Numerical Solution of Retarded and Neutral Delay Differential Equations using Continuous Runge-Kutta Methods

by

Hiroshi Hayashi

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Numerical Solution of Retarded and Neutral Delay Differential Equations using Continuous Runge-Kutta Methods

Hiroshi Hayashi

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy Graduate Department
of Computer Science, University of Toronto, 1996

Abstract

A delay differential equation (DDE) can provide us with a realistic model of many phenomena arising in applied mathematics. For example, a DDE can be used for the modeling of population dynamics, the spread of infectious diseases, and two-body problems of electrodynamics. In this thesis, we present a numerical method for solving DDEs and analysis for the numerical solution.

We have developed the method by adapting recently developed techniques for initial value ordinary differential equations (continuous Runge-Kutta formulas, defect error control, and an automatic handling technique for derivative discontinuities) and developing a new approach (that is based on an iterative scheme determined by extrapolation or a special interpolant) to handle vanishing delays. This results in a robust variable stepsize method which can be applied to problems with state dependent delays. This approach can also be applied to any continuous Runge-Kutta formula.

We determine convergence properties for the numerical solution associated with our method. We first analyze such properties for retarded type DDEs and then the analysis is extended to the case of neutral type DDEs. The main theoretical result we establish is that the global error of the numerical solution is bounded by a multiple of the prescribed tolerance. Our analysis can also be applied to other numerical methods based on Runge-Kutta formulas for DDEs.

We have developed an experimental Fortran code as a modified version of DVERK based on a 6th order continuous Runge-Kutta formula. An implementation of our method is described and numerical results for various kinds of DDEs are presented. Numerical comparisons with two other existing codes, ARCHI and DRKLAG, are considered and some advantages of our approach are identified.

Acknowledgments

I would like to express my sincerest appreciation to my supervisor, Professor Wayne Enright, for giving me the opportunity to pursue my Ph.D degree at the University of Toronto, and for his constant encouragement and advice, which always led me to the right research direction.

I would like to thank the members in my thesis committee, Professor Christina Christara, Professor Simon Fraser, Professor Ken Jackson, Professor Allan Jepson, and Professor Rudi Mathon, for reading manuscripts and providing numerous suggestions. In particular, I am grateful to Professor Ken Jackson for his careful reading and valuable suggestions. I would also like to thank my external examiner Professor Christopher Baker of the University of Manchester for his valuable suggestions and criticism.

I wish to thank Dr. Edisanter Lo, Dr. Christopher Paul, Dr. Dexter Sutherland, Professor Skip Thompson, and Professor Rémi Vaillancourt for providing their programs with their related papers.

I am grateful to Dr. Tom Fairgrieve, Mr. Arunkumar Jonnalagadda, and Dr. Hung Nguyen for reading the early drafts of this thesis and giving me useful comments.

I am grateful to my former supervisors, Professor Taketomo Mitsui and Professor Tatsuo Torii of Nagoya University, for their continuing support.

I wish to thank the University of Toronto and the Department of Computer Science for their generous financial support.

Finally I would like to thank my wife Mio for her moral support and understanding. Without her encouragement, this thesis would never been completed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

A delay differential equation (DDE) can provide us with a realistic model of many phenomena arising in applied mathematics. For example, a DDE can be used for the modeling of population dynamics, the spread of infectious diseases, two-body problems of electrodynamics, etc. More examples are discussed in (Driver 1977, pp. 226–240), (Hairer, Nørsett and Wanner 1993, pp. 339–353), (Kuang 1993, pp. 3–11), and (Macdonald 1989, pp. 9–27).

Although there exist several codes to solve DDEs, most of them are concerned only with retarded type[1] DDEs and cannot handle some difficulties properly. At the present time, as far as we know, only two research codes exist to solve both retarded and neutral type[1] DDEs with vanishing delays[1]: ARCHI (Paul 1992b), developed by Baker and Paul, DRKLAG (Neves and Thompson 1992a), developed by Neves and Thompson. This fact together with the recent development of continuous numerical methods and some other tools useful in solving DDEs motivated us to work on this subject.

In this thesis, we develop a numerical method for retarded and neutral type DDEs and present a convergence analysis for the numerical solution. In this chapter, we give assumptions and definitions of terminologies that will be used in this thesis. We then address numerical difficulties when an existing ordinary differential equation (ODE) solver is adapted to solve DDEs. Previous work on numerical methods for DDEs is reviewed. Contributions and an outline of this thesis are then described.

[1]See §1.2.1 for the definitions.

1.2 Background

1.2.1 Assumptions and Definitions

An initial value problem for ODEs without delays is often referred to in this thesis. To distinguish it from an initial value problem for DDEs, we call the following initial value problem an SDE (abbreviated from *S*tandard O*DE*).

$$\begin{aligned} y'(t) &= f(t, y(t)), \quad \text{for} \quad t_0 \le t \le t_F, \\ y(t_0) &= y_0, \end{aligned} \tag{1.1}$$

where t is an independent variable, $'$ represents the right-hand derivative with respect to t, y and f are $\mathcal{N}$-vector functions, and y_0 is an $\mathcal{N}$-vector constant.

We will only consider systems of first order DDEs. A DDE refers to both a retarded type DDE (RDE) and a neutral type DDE (NDE). A RDE is an ODE involving solution delay terms and given by

$$\begin{aligned} y'(t) &= f(t, y(t), y(t-\sigma_1(t, y(t))), y(t-\sigma_2(t, y(t))), \\ &\qquad \cdots, y(t-\sigma_\nu(t, y(t)))), \quad \text{for} \quad t_0 \le t \le t_F, \\ y(t) &= \phi(t), \quad \text{for} \quad t \le t_0, \end{aligned} \tag{1.2}$$

where y, f, and ϕ are $\mathcal{N}$-vector functions and $\sigma_i, i = 1, 2, \ldots, \nu$ are scalar functions. An NDE is an ODE involving solution and derivative delay terms and give by

$$\begin{aligned} y'(t) &= f(t, y(t), y(t-\sigma_1(t, y(t))), y(t-\sigma_2(t, y(t))), \\ &\qquad \cdots, y(t-\sigma_\nu(t, y(t))), y'(t-\sigma_{\nu+1}(t, y(t))), \\ &\qquad \cdots, y'(t-\sigma_{\nu+\omega}(t, y(t)))) \quad \text{for} \quad t_0 \le t \le t_F, \\ y(t) &= \phi(t), \quad y'(t) = \phi'(t), \quad \text{for} \quad t \le t_0, \end{aligned} \tag{1.3}$$

where y, f, and ϕ are $\mathcal{N}$-vector functions and $\sigma_i, i = 1, 2, \ldots, \nu + \omega$ are scalar functions. For simplicity, we assume $\nu = 1$ for RDEs, and $\nu = 0, \omega = 1$ for NDEs without loss of generality. Thus, we consider the RDE

$$\begin{aligned} y'(t) &= f(t, y(t), y(t-\sigma(t, y(t)))), \quad \text{for} \quad t_0 \le t \le t_F, \\ y(t) &= \phi(t), \quad \text{for} \quad t \le t_0, \end{aligned} \tag{1.4}$$

and the NDE

$$\begin{aligned} y'(t) &= f(t, y(t), y'(t - \sigma(t, y(t)))), \quad \text{for} \quad t_0 \le t \le t_F, \\ y(t) &= \phi(t), \quad y'(t) = \phi'(t), \quad \text{for} \quad t \le t_0, \end{aligned} \tag{1.5}$$

where y, f, and ϕ are $\mathcal{N}$-vector functions and σ is a scalar function. An extension of results presented in Chapters 2 and 3 to (1.2) and (1.3) is straightforward and our experimental Fortran code has been developed to solve these general forms.

The abbreviations, DDE, SDE, RDE, and NDE are used to refer to the differential equation alone or to the differential equation plus the associated initial condition (the meaning will be clear from the context).

We call the function $\sigma(t, y(t))$ the *delay*, $t - \sigma(t, y(t))$ the *delay argument*, the value of the solution delay term $y(t - \sigma(t, y(t)))$ the *(solution) delay value* and the value of the derivative delay term $y'(t - \sigma(t, y(t)))$ the *derivative delay value.* If the delay is a constant, it is called the *constant delay.* If it is a function of only time, then it is called the *time dependent delay.* If the delay is a function of time t and the solution $y(t)$, it is called the *state dependent delay.* A delay argument that passes the current time (*i.e.*, $t - \sigma(t, y(t)) > t$) is called an *advanced delay.* We do not consider the latter type of a DDE in this thesis. Furthermore, for the approximation $z(t)$, we also assume that $t - \sigma(t, z(t)) \le t$.

When $\sigma(t, y(t)) = 0$ for the NDE (1.5), we assume that the matrix

$$\Big[I - \frac{\partial f}{\partial y'}\Big]\Big|_{t, y(t), y'(t)}, \tag{1.6}$$

is full rank so that $y'(t)$ is uniquely determined as a function of t and $y(t)$ at the point where $\sigma(t, y(t)) = 0$.

We also assume the existence, uniqueness, and stability of solutions to the problems under consideration. For example, sufficient conditions for the existence and uniqueness of solutions to the RDE (1.4) are f is continuous with respect to $t, y(t)$, and $y(t - \sigma(t, y(t)))$, $y(t)$ is continuous, f satisfies a Lipschitz condition in the last two arguments, ϕ is continuous, and f is bounded (Driver 1977). For the NDE (1.5), it may be required that a Lipschitz constant for the derivative delay term be less than one (Grimm 1971) as well as similar conditions to those for the RDE. This restriction on the Lipschitz constant ensures that the matrix of (1.6) is full rank on the whole interval.

A function f that satisfies an inequality

$$\|f(t, y_2, z_2) - f(t, y_1, z_1)\| \leq L_1\|y_2 - y_1\| + L_2\|z_2 - z_1\|, \tag{1.7}$$

for all $(t, y_1, z_1), (t, y_2, z_2)$ in a region D is said to satisfy a *Lipschitz condition* in D. We use a symbol L_3 for a Lipschitz constant corresponding to the third argument if a problem under consideration is an NDE. We use symbols L_y and $L_{y'}$ for Lipschitz constants for the functions y and y' in inequalities

$$\begin{aligned} \|y(t_2) - y(t_1)\| &\leq L_y\|t_2 - t_1\|, && (1.8)\\ \|y'(t_2) - y'(t_1)\| &\leq L_{y'}\|t_2 - t_1\|. && (1.9) \end{aligned}$$

and L_σ for a Lipschitz constant for the function σ in an inequality

$$\|\sigma(t, y_2) - \sigma(t, y_1)\| \leq L_\sigma\|y_2 - y_1\|. \tag{1.10}$$

Let $y(t)$ be a solution of a differential equation and $z_i(t)$ be a continuous approximation to $y(t)$ on $[t_i, t_{i+1}]$ associated with a method. The method is said to be *convergent* if

$$\max_{0\leq i\leq N} \max_{t_i\leq t\leq t_{i+1}} \|z_i(t) - y(t)\| \to 0 \quad \text{as} \quad H = \max_i h_i \to 0 \quad \text{and} \quad N \to \infty,$$

where $h_i = t_{i+1} - t_i$ and $\sum_{i=0}^{N} h_i = t_F - t_0$.

We define a *local solution* of the RDE (1.4) as the solution of

$$\begin{aligned} y_n'(t) &= f(t, y_n(t), y_n(t - \sigma(t, y_n(t)))), \quad \text{for} \quad t_n \leq t \leq t_{n+1}, \\ y_n(t) &= z(t), \quad \text{for} \quad t \leq t_n, \end{aligned} \tag{1.11}$$

where $z(t)$ is a continuous approximation to $y(t)$ on $(t_0, t_n]$ associated with a method and $z(t) = \phi(t)$ for $t \leq t_0$. A local solution of the NDE (1.5) is defined as the solution of

$$\begin{aligned} y_n'(t) &= f(t, y_n(t), y_n'(t - \sigma(t, y_n(t)))), \quad \text{for} \quad t_n \leq t \leq t_{n+1}, \\ y_n(t) &= z(t), \quad y_n'(t) = z'(t), \quad \text{for} \quad t \leq t_n. \end{aligned} \tag{1.12}$$

A method is said to be of *order*[2] p if the local error is of order $p+1$, *i.e.*,

$$\|y_n(t_n+\tau h_n)-z_n(t_n+\tau h_n)\| = O(h_n^{p+1}),$$

for all $\tau \in [0,1]$. Here $z_n(t)$ is the continuous approximation to $y(t)$ on $[t_n, t_{n+1}]$ associated with the method.

A method for the RDE (1.4) is said to be *consistent* if the method satisfies

$$z_n(t_n+\tau h_n) = z(t_n) + \tau h_n f(t_n, z(t_n), z(t_n-\sigma(t_n, z(t_n)))) + O(h_n^2), \tag{1.13}$$

for $\tau \in [0,1]$. For the NDE (1.5), a method is consistent if

$$z_n(t_n+\tau h_n) = z(t_n) + \tau h_n f(t_n, z(t_n), z'(t_n-\sigma(t_n, z(t_n)))) + O(h_n^2). \tag{1.14}$$

1.2.2 Numerical Difficulties

A conventional approach for solving DDEs is to adapt an SDE solver. We follow this approach by adapting an SDE solver based on a continuous Runge-Kutta formula. There are two major numerical difficulties to be addressed when using this approach: discontinuities may occur in various orders of the derivative of the solution; the delay may fall in the current step because it is smaller than the stepsize or may even vanish, *i.e.*, $\sigma \to 0$. We call the second difficulty a *small delay.* Furthermore when the delay vanishes, we call it a *vanishing delay.*

The first difficulty is due to the presence of the delay term and a discontinuity can propagate from the initial interval. A discontinuity causes numerical difficulty because numerical methods for SDEs are usually developed by assuming sufficient smoothness of solutions. To illustrate this difficulty, consider the problem

$$\begin{array}{rcll} y'(t) & = & y(t-1), & \text{for} \quad 0 \le t, \\ y(t) & = & \phi(t) \ = \ 1, & \text{for} \quad t \le 0. \end{array}$$

[2]The order of a method is usually associated with the order of the global error obtained with fixed stepsize. In this thesis, we relate a global error bound to the size of the defect and we do not consider directly the order of the global error. As defect is a local property, we define the order of method as above.

The left- and the right-hand derivative do no agree at $t = 0$, since

$$(\phi^{'}(0) = 0) \neq (y'(0) = 1).$$

This propagates to $t = 1, 2, 3, \cdots$, although the order of the derivative discontinuity[3] increases as t increases. For example, the left- and the right-hand derivatives of y at $t = 1$ are

$$\begin{aligned} \lim_{h \to 1^-} y'(h) &= \lim_{h \to 1^-} y(h-1) = 1, \\ \lim_{h \to 1^+} y'(h) &= \lim_{h \to 1^+} y(h-1) = 1, \end{aligned}$$

while the left- and the right-hand second derivatives of y at $t = 1$ are

$$\begin{aligned} \lim_{h \to 1^-} y^{''}(h) &= \lim_{h \to 1^-} y'(h-1) = 0, \\ \lim_{h \to 1^+} y^{''}(h) &= \lim_{h \to 1^+} y'(h-1) = 1. \end{aligned}$$

Note that the solution of this problem on $[0, 1]$ is $y(t) = t + 1$ (El'sgol'ts and Norkin 1973, pp. 6–15). In general, the order of a derivative discontinuity increases with t for RDEs, but this is not usually true for NDEs. This non-smoothness of a derivative discontinuity for NDEs causes a numerical difficulty especially when the delay vanishes in the integration interval. In this case, there may be a cluster of low order derivative discontinuities in the neighborhood of the vanishing point. It would not be appropriate to locate such discontinuity points by a code since the stepsize would have to be smaller than the delay near the vanishing point. For this difficulty, the order of any Runge-Kutta formulas will be reduced.

The second difficulty is easily understood after the concept of a continuous Runge-Kutta formula is introduced. Thus, this difficulty will be discussed in the next chapter along with our approach to resolve the difficulty.

[3] An order of derivative discontinuity for a function of t is said to be r at $t = t_d$ if its lth left-hand derivative and lth right-hand derivative agree at $t = t_d$ for $l = 0, \ldots, r-1$ while its rth left-hand derivative and rth right-hand derivative do not agree at $t = t_d$.

1.3 A Review of Previous Work

1.3.1 Numerical Methods for RDEs

Tavernini (Tavernini 1971) proposed a one step method of arbitrary order p for Volterra integro-differential equations. (The Volterra integro-differential equation Tavernini considered is

$$y'(t) = f(t, y(t), \int_{t_0}^{t} k(y(s), s, t)\ ds),$$

where the *kernel* k is a function depending on the solution $y(t)$ over $[t_0, t]$.) Suppose a continuous approximation $z(t)$ to $y(t)$ is computed on $[t_0, t_n]$, where $z(t)$ is a piecewise polynomial $z_i(t)$ defined on $[t_i, t_{i+1}]$ for $i = 0, 1, \cdots, n-1$. Then the continuous approximation $z_n(t)$ on $[t_n, t_{n+1}]$ for the RDE (1.4) is constructed by Tavernini's method as follows. First, an initial approximation is given by the formula

$$z_n^{(0)}(t_n + \tau h_n) = z(t_n) + \tau h_n f(t_n, z(t_n), z(t_n - \sigma)),$$

for $0 \le \tau \le 1$. Then the qth approximation is given by the formula

$$z_n^{(q)}(t_n + \tau h_n) = z(t_n) + h_n \sum_{i=0}^{q} b_i^{(q)}(\tau) f(t_n + c_i^{(q)} h_n, z_n^{(q-1)}(t_n + c_i^{(q)} h_n), z_n^{(q-1)}(t_n + c_i^{(q)} h_n - \sigma)),$$

for $q = 1, 2, \cdots, p$. Finally, the approximation $z_n(t)$ is defined by the pth approximation $z_n^{(p)}(t)$. If the delay argument is not in the current step $[t_n, t_{n+1}]$, then the approximation $z(t)$ is used to evaluate the delay term instead. The coefficients $c_i^{(q)}$ are chosen in $[0, 1]$. To determine the functions $b_i^{(q)}(\tau)$, algebraic systems must be solved so that the condition

$$\|z_n^{(q)}(t) - y_n(t)\| = O(h_n^{q+1}), \quad \text{for} \quad t_n \le t \le t_{n+1},$$

is satisfied, where $y_n(t)$ is the local solution. It has been pointed out by Jackiewicz (Jackiewicz 1981b) that the order of this method extended for NDEs does not necessarily exceed one.

Neves (Neves 1975b) developed a numerical method based on the fourth order Merson's formula with a two-point Hermite interpolant defined by the endpoints of each step. The Fortran code DMRODE is presented in (Neves 1975a). Neves (Neves 1975b) has suggested two ways to handle discontinuities with the code: ignoring them and depending on the error control mechanism; stopping the integration at the discontinuity point and then requiring the user to restart the integration. The method proposed by Tavernini is used in DMRODE when a delay is smaller than the current stepsize. Neves (Neves 1981) has also proposed two error control mechanisms which handle discontinuities: monitoring the difference between the delay value computed by extrapolation from the previous step and the delay value computed by a local interpolant; monitoring the difference between the solution computed by extrapolation from the previous step and the solution computed by a local interpolant. Neves and Feldstein (Neves and Feldstein 1976) discussed the characteristics of discontinuities for DDEs with state dependent delays. They pointed out that the order of the derivative discontinuity for RDEs will generally smooth as time increases while there may be no such smoothing for NDEs. An automatic detection of discontinuities by means of a nonlinear equation solver is described by Feldstein and Neves (Feldstein and Neves 1984).

Oppelstrup (Oppelstrup 1978) described a numerical method based on the Runge-Kutta Fehlberg 5(4) formula with a three-point fourth-degree Hermite-Birkhoff interpolant. This method does not directly handle discontinuities by some special techniques.

Oberle and Pesch (Oberle and Pesch 1981) developed a numerical method for RDEs with constant delays. The method is based on the Runge-Kutta Fehlberg 4(5) and 7(8) pairs with three- and five-points Hermite interpolants. These mesh points are assumed to be chosen such that they satisfy the following three conditions: they do not cross discontinuities; extrapolation is avoided; the delay argument under consideration should be close to the center of the mesh points chosen. Propagated derivative discontinuity points are determined at the beginning of integration. (Only constant delay is considered.)

Arndt (Arndt 1984) developed a general convergence theory for RDEs by considering that a numerical method is actually trying to solve the SDE

$$
\begin{aligned}
\bar{y}'(t) &= f(t, \bar{y}(t), z(t-\sigma(t, z(t)))), \quad \text{for} \quad t_0 \le t \le t_F, \\
\bar{y}(t_0) &= y_0, \\
(\bar{y}(t) &= \phi(t), \quad \text{for} \quad t \le t_0,)
\end{aligned}
$$

where $z(t)$ is a known continuous approximation to $\bar{y}(t)$. This approximation $z(t)$ is known before the numerical solution is computed in each step unless the delay argument is in the current step.

Neves and Thompson (Neves and Thompson 1992a; Neves and Thompson 1992b) developed further the ideas used in DMRODE. Their method is based on the continuously embedded Runge-Kutta formula pair 5(4) of Sarafyan with a cubic 4th order interpolant. The resulting Fortran code DRKLAG is described in (Neves and Thompson 1992b). Discontinuity difficulties are handled either by the error control mechanism or locating the discontinuity points by solving associated nonlinear equations. Small delays and vanishing delays are handled by either restricting the stepsize to be smaller than the delays or using extrapolation. When extrapolation is used, the amount the stepsize may increase on each step is restricted so that extrapolation is sufficiently accurate. In the case where a delay is smaller than the stepsize in the first step, DRKLAG uses the approach proposed by Tavernini.

Willé and Baker (Willé and Baker 1992) discussed a technique to locate propagated derivative discontinuities by means of a directed graph. With the use of such a graph, the dependency of propagated discontinuities among the components of RDEs are specified. This is particularly important for systems of equations with multiple delays.

Baker and Paul (Baker and Paul 1993a; Baker and Paul 1993b) and Paul (Paul 1992a) developed a numerical method based on a continuous Runge-Kutta method with an iterative scheme for vanishing delays. Suppose the continuous approximation $z(t)$ is computed on $[t_0, t_n]$, where $z(t)$ is a piecewise polynomial $z_i(t)$ defined on $[t_i, t_{i+1}]$ for $i = 0, 1, \cdots, n-1$. Then the continuous approximation $z_n(t)$ on $[t_n, t_{n+1}]$ for the RDE (1.4) is constructed by the method as follows. Assume that the delay arguments for all stages are in the current step. Then the qth continuous approximation to the solution is given by the formula

$$\begin{aligned} z_n^{(q)}(t_n + \tau h_n) = \\ & z(t_n) + h_n \sum_i b_i(\tau) f(t_n + c_i h_n, z_n^{(q-1)}(t_n + c_i h_n), z_n^{(q-1)}(t_n + c_i h_n - \sigma(t, z_n^{(q-1)}(t_n + c_i h_n)))), \end{aligned}$$

for $0 \leq \tau \leq 1$ and $q = 1, 2, \cdots, m$, where m is the number of iterations. The initial approximation $z_n^{(0)}(t_n + c_i h_n - \sigma(t, z_n^{(0)}(t_n + c_i h_n)))$ to the delay term is given by either the approximation at t_n or extrapolation from the previous step and the initial approximation $z_n^{(0)}(t_n + c_i h_n)$ to the solution term are given by the formula

$$z_n^{(0)}(t_n + c_i h_n) = z(t_n) + h_n \sum_j^{i-1} a_{ij} f(t_n + c_j h_n, z_n^{(0)}(t_n + c_j h_n), z_n^{(0)}(t_n + c_j h_n - \sigma(t, z_n^{(0)}(t_n + c_i h_n)))).$$

Then an approximation $z_n(t)$ is defined by the mth approximation $z_n^{(m)}(t)$. If some delay arguments are in the past steps, the continuous approximation $z(t)$ is used to compute the delay values. The Fortran code ARCHI (Paul 1992b) based on this method is described in Chapter 4.

Karoui and Vaillancourt (Karoui and Vaillancourt 1994) developed a numerical method based on the Verner's 5(6) Runge-Kutta pair with a three-point fifth-degree Hermite interpolant. For discontinuity handling, the method checks the sign of an associated event function at both endpoints of each step and if the sign is different, a fifth-degree polynomial is constructed using data on six previous mesh points and the discontinuity point is located by using bisection. This method, implemented as a Fortran code SYSDEL, has been extended so that small delays are handled by extrapolation in (Karoui and Vaillancourt 1995).

There are also at least two other Fortran codes which solve RDEs. One is DDELAY based on Adams-Moulton methods and Gear's BDF methods by Kahaner and Sutherland. DDELAY solves both RDEs and NDEs, but does not solve a problem if its delay vanishes except at the initial point. RETARD is another code based on the Dormand-Prince 5(4) pair DOPRI5 by Hairer, Nørsett, and Wanner (Hairer, Nørsett and Wanner 1993). This only solves RDEs and also cannot handle small delays.

1.3.2 Numerical Methods for NDEs

NDEs have received little attention in the literature although the codes ARCHI, DDELAY, and DRKLAG have been developed. This might be because NDEs are hard to analyze mathematically: there is no smoothing in propagated derivative discontinuities and one must approximate derivative delay terms.

Two implicit one step methods of arbitrary order p were proposed by Jackiewicz (Jackiewicz 1984) for the NDE

$$y'(t) = f(t, y(t-\sigma_1), y'(t-\sigma_2)),$$

based on the Picard iteration

$$y^{(q)}(t) = y(t_n) + \int_{t_n}^{t} z^{(q-1)}(s)\ ds,$$

for $q = 1, 2, \cdots, p$. The qth approximation $z^{(q)}$ is defined by either

$$z^{(q)}(t) = f\Big(t, y(t_n) + \int_{t_n}^{t-\sigma_1} z^{(q-1)}(s)\ ds, z^{(q)}(t-\sigma_2)\Big),$$

or

$$z^{(q)}(t) = f\Big(t, y(t_n) + \int_{t_n}^{t-\sigma_1} z^{(q)}(s)\ ds, z^{(q)}(t-\sigma_2)\Big).$$

Then the integral is replaced by a quadrature formula of sufficiently high order.

Jackiewicz also has published a series of papers investigating the solution of NDEs by multistep methods: methods in (Jackiewicz 1981a; Jackiewicz 1982) are based on Adams-Moulton formulas; methods in (Jackiewicz 1986) are based on predictor-corrector formulas with fixed order; methods (Jackiewicz 1987) are based on variable-step variable-order Adams-Bashford Adams-Moulton predictor-corrector formulas with equal intervals for back values; methods in (Jackiewicz and Lo 1991) are based on variable-step variable-order Adams-Bashford Adams-Moulton predictor-corrector formulas with unequal intervals for back values.

1.4 Contributions of the Thesis

The main contribution of this thesis is the development and analysis of a numerical method for solving retarded and neutral type DDEs with vanishing delays. The method has been developed by adapting recently developed techniques[4] for SDEs (continuous Runge-Kutta formulas, defect error control, and automatic handling for discontinuities) and developing a new approach (that is based on an iterative scheme determined by extrapolation or a special interpolant) to handle vanishing delays. Our method is unique in the sense that the defect (perturbation) in the perturbed differential equation

$$z'(t) = f(t, z(t), z(t-\sigma(t, z(t)))) + \delta(t),$$

is directly controlled. The automatic handling technique for discontinuities turns out to be effective and important for DDEs because, otherwise, the user must supply various information to solve DDEs. For example, ARCHI requires the user to supply the dependency of derivative discontinuities among components of a system, the location of any derivative discontinuities in the initial interval, and the order of the derivative discontinuities. Our method can handle vanishing delay problems and is less expensive than other methods for vanishing delay problems.

We determine convergence properties for the numerical solution associated with our method. We first analyze such properties for RDEs and then the analysis is extended to the case of NDEs.

[4] They are reviewed in Chapter 2.

This extension is especially important since NDEs have received little attention in the literature. The main theoretical result we establish is that the global error of the numerical solution is bounded by a multiple of the prescribed tolerance.

We have implemented our method as an experimental Fortran code based on DVERK (Hull, Enright and Jackson 1976). It is worth noting that only two research codes (ARCHI and DRKLAG) can handle RDEs and NDEs with vanishing delays. We have carried out numerical experiments over various kinds of DDEs and reported the statistics. We have also reported the statistics for ARCHI and DRKLAG for comparison purpose. We discuss the performance of three codes in terms of cost and reliability and identify some advantages of our method.

1.5 An Outline of the Thesis

In Chapter 2, we review continuous Runge-Kutta formulas, defect control, the automatic handling technique for discontinuities and then discuss how we adapt these techniques to solve DDEs. We then describe our algorithms for computing small/vanishing delays in detail. In Chapter 3, sufficient conditions for convergence of the numerical solution, the order of the defect, the global error bound for RDEs are first investigated. Similar analysis is then carried out for NDEs. In Chapter 4, the description of the Fortran code we have implemented is given, numerical experiments for various kinds of DDEs are presented, and numerical comparisons with ARCHI and DRKLAG are considered. In Chapter 5, a summary of this thesis is presented and future investigations are discussed.

Chapter 2

Numerical Methods

2.1 An Explicit Continuous Runge-Kutta Formula for DDEs

An s-stage explicit Runge-Kutta (RK) formula for the SDE (1.1) to compute a numerical solution at $t_n + h_n$ is defined by

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

for $i = 1, 2, \cdots, s$, where a_{ij}, b_i, c_i are the coefficients of the Runge-Kutta formula and y_n is the associated approximation to $y(t_n)$. The Runge-Kutta formula is usually represented by a tableau

$$\begin{array}{c|ccccc} 0 & 0 & & & & \\ c_2 & a_{21} & 0 & & & \\ \vdots & & & & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

This class of Runge-Kutta formulas produces approximations only at discrete points. For some applications, a continuous approximation to the solution of an SDE is needed, for example, to obtain dense output for plotting the solution of an SDE, to find roots of a function associated with the solution of an SDE, or to handle discontinuities (Enright et al. 1988)(Hairer, Nørsett and Wanner 1993, pp. 188–203). Consequently, continuous Runge-Kutta (CRK) formulas, which produce a continuous approximation to the solution of an SDE on $[t_0, t_F]$, have been developed (Dormand and Prince 1986; Enright et al. 1986; Gladwell et al. 1987; Horn 1983; Shampine 1985; Shampine 1986; Verner 1993) and have the form

$$\begin{aligned} z_n(t_n + \tau h_n) &= z_{n-1}(t_n) + h_n \sum_{i=1}^{\bar{s}+1} \bar{b}_i(\tau) k_i, \\ k_i &= f(t_n + c_i h_n, Y_i), \\ Y_i &= z_{n-1}(t_n) + h_n \sum_{j=1}^{i-1} a_{ij} k_j, \end{aligned}$$

for $i = 1, 2, \cdots, \bar{s}+1$ and $\tau \in [0,1]$, where $\bar{b}_1(\tau), \bar{b}_2(\tau), \cdots, \bar{b}_{\bar{s}+1}(\tau)$ are polynomials with respect to τ and $z_{-1}(t_0) = y_0$.

A tableau for the restricted class of CRK formulas we consider is shown below.

$$\begin{array}{c|cccccc} 0 & 0 & & & & & \\ c_2 & a_{21} & 0 & & & & \\ \vdots & & & & & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & 0 & \\ 1 & b_1 & b_2 & \cdots & b_{s-1} & b_s & 0 \\ \vdots & & & & & & \\ c_{\bar{s}+1} & a_{\bar{s}+1,1} & a_{\bar{s}+1,2} & \cdots & & a_{\bar{s}+1,\bar{s}} & 0 \\ \hline & \bar{b}_1(\tau) & \bar{b}_2(\tau) & \cdots & & \bar{b}_{\bar{s}}(\tau) & \bar{b}_{\bar{s}+1}(\tau) \end{array}$$

This class of CRK formula is constructed by adding extra stages $s+1$ to $\bar{s}+1$ with $c_{s+1} = 1, a_{s+1,j} = b_j$ for $j = 1, \ldots, s$ so that the approximation can easily be developed as a C^1 interpolant. Note that since $k_{s+1} = f(t_{n+1}, z_n(t_{n+1}))$ will be the first stage on the next attempted step, the actual cost per successful step is $\bar{s}$ derivative evaluations. (If t_{n+1} is a discontinuity point, this relation is not satisfied. Thus, k_1 must be computed on $[t_{n+1}, t_{n+2}]$.)

Since a CRK method can more easily handle some difficulties which DDEs inherit, *i.e.*, the necessity of evaluations of the delay term at off-mesh points and the ability to cope with discontinuities (to be explained in the next section), we have adopted a CRK formula as the basis for our approach to DDEs. We define a CRK formula for the RDE (1.4) by

$$
\begin{aligned}
z_n(t_n+\tau h_n) &= z_{n-1}(t_n)+h_n\sum_{i=1}^{\bar{s}+1}\bar{b}_i(\tau)k_i, \\
k_i &= f(t_n+c_ih_n, Y_i, z(t_n+c_ih_n-\sigma(t_n+c_ih_n, Y_i))), \\
Y_i &= z_{n-1}(t_n)+h_n\sum_{j=1}^{i-1}a_{ij}k_j,
\end{aligned}
\tag{2.1}
$$

for $i=1,2,\cdots,\bar{s}+1$, where $z_{-1}(t_0)=\phi(t_0)$. Here $z(t)$ is the associated continuous approximation to the solution of (1.4) used to determine an approximation to the required delay term with $z(t)=\phi(t)$ for $t\leq t_0$.

In order to approximate the delay term on $[t_0,t_n]$ by evaluating a piecewise polynomial $z_j(t)$ for $j=0,1,\cdots,n-1$ when needed, we save the solutions and derivatives at the endpoints of each step together with some internal derivative values. The internal representation of the polynomial interpolant for evaluating the delay term is

$$
\begin{aligned}
z_j(t_j+\tau h_j) &= d_{j,0}(\tau)z_j(t_j)+d_{j,1}(\tau)h_jz_j'(t_j)+d_{j,2}(\tau)z_j(t_j+h_j) \\
&\quad + d_{j,3}(\tau)h_jz_j'(t_j+h_j)+h_j\textstyle\sum_{l=s+2}^{\bar{s}+1}d_{j,l-s+2}(\tau)k_l,
\end{aligned}
\tag{2.2}
$$

where the $d_{j,k}(\tau)$ are polynomials with respect to τ. See (Enright et al. 1986) for the derivation of this representation. One reason to use the above representation rather than (2.1) is that the above representation usually needs less storage[1]. Another reason is related to a discontinuous problem and will be discussed in the next section. An approach we develop for the case where delay arguments for some stages are in the current step, $[t_n,t_{n+1}]$, (and hence the equation (2.1) becomes implicit) is described in §2.3.

We use the internal stage value, Y_i, for computing the ith delay argument associated with k_i for state dependent delay problems. Since the internal stage value of an explicit (C)RK formula, Y_i, is usually a low order approximation to the local solution, the continuous approximation $z(t)$ interpreted as an approximation to Y_i will be of low order regardless of how well $z(t)$ approximates the local solution. Nevertheless it will be proved in the next chapter that this low order approximation to the delay arguments does not affect the order of the approximation to the delay value or the overall order of the discrete or continuous numerical solution. One may use the continuous approximation $z(t)$ to compute the delay argument instead of Y_i. However,

[1] $s=7$ and $\bar{s}=10$ for the CRK formula we use.

this makes the equation (2.1) implicit. In addition to that the known theory for explicit RK methods (applied in Chapter 3) cannot be applied if we use the continuous approximation $z(t)$.

The adoption of the CRK formula (2.1) to the NDE (1.5) is accomplished simply by replacing the continuous approximation $z(t)$ by its derivative to evaluate the derivative delay term. The CRK formula for the NDE is defined, analogous to (2.1), by

$$\begin{aligned} z_n(t_n+\tau h_n) &= z_{n-1}(t_n) + h_n \sum_{i=1}^{\bar{s}+1} \bar{b}_i(\tau) k_i, \\ k_i &= f(t_n + c_i h_n, Y_i, z'(t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i))), \\ Y_i &= z_{n-1}(t_n) + h_n \sum_{j=1}^{i-1} a_{ij} k_j, \end{aligned} \tag{2.3}$$

for $i = 1, 2, \cdots, \bar{s}+1$, where $z_{-1}(t_0) = \phi(t_0)$ and $z'(t) = \phi'(t)$ for $t \le t_0$.

2.2 Defect Control and Handling Discontinuities

A *defect* is a measure of the amount by which a continuous numerical solution fails to satisfy the differential equation. The defect for RDEs on the step $[t_n, t_{n+1}]$ is defined by

$$\delta_n(t) = z_n'(t) - f(t, z_n(t), z(t - \sigma(t, z_n(t)))),$$

where $z_n(t)$ is defined by the equation (2.1). The defect for NDEs on the step $[t_n, t_{n+1}]$ is defined by

$$\delta_n(t) = z_n'(t) - f(t, z_n(t), z'(t - \sigma(t, z_n(t)))),$$

where $z_n(t)$ is defined by the equation (2.3). Here $z(t) = z_i(t)$ if $t_i \le t \le t_{i+1}$ for $i = 0, 1, \cdots, n$ and $z(t) = \Phi(t)$ if $t \le t_0$. The method we develop attempts to ensure that

$$\max_{0 \le \tau \le 1} \|\delta_n(t_n + \tau h_n)\| \le \mathrm{TOL},$$

on each step for user supplied tolerance TOL. This error control mechanism is called *defect control* (Enright 1989a; Enright 1989b; Enright 1993; Enright and Higham 1991; Higham 1989a; Higham 1989b; Higham 1989c; Higham 1989d) in contrast to local error control.

Defect control quantifies the error in the numerical solution over the whole interval while standard local error control quantifies the discrete error associated with the mesh points. Another advantage over local error control discussed in (Enright 1989b) is that the relationship between achieved accuracy and prescribed tolerance is almost independent of the method while this is not usually true for local error control (especially for RK methods).

It is natural to use defect control for DDEs since a continuous approximation is always available. We use a defect associated with a single sample point in each step as an estimate for the maximum defect; thus, only one extra function evaluation is needed for the error control and the stepsize selection mechanism. Defect control for SDEs based on a single sample defect value is discussed in (Enright 1989b). One might feel that a single sample point in $[t_n, t_{n+1}]$ may not be appropriate for DDEs because of the delay argument which is possibly in a past step. However, by the definition of local problems (1.11) and (1.12), the approximation $z(t)$ in a past step is the exact local solution. Thus, a similar argument about a single sample point defect control for SDEs can be applied to DDEs. We demonstrate in Chapter 4 how well a single sample point defect control works by computing the defect at 100 sample points on each step over a wide class of test problems. For the problems we solved, the maximum observed defect is found to be less than 10·TOL on 90 % of the steps.

In order to observe an optimal order when applying a RK formula to an SDE, we must make sure that the solution of the SDE is sufficiently smooth on each subinterval because the RK formula is based on the fact that a series expansion of the RK formula agrees with that of the solution up to the prescribed order. In addition to the same type of discontinuities that can arise in the solution of SDEs, we must consider propagated discontinuities described in §1.2.2.

The strategy we adopt to cope with discontinuities is based on the method for discontinuous SDEs developed and discussed in detail in (Enright et al. 1988). After each accepted attempted step from t_n to $t_n + h$ which was immediately preceded by a rejected step from t_n to $t_n + H$, we check whether the next proposed stepsize is large due to a very small (estimated) defect in the accepted attempted step. When this is the case, we suspect that a discontinuity may exist on $(t_n + h, t_n + 2h]$. (In our code, the convention used is $h < H \leq 2h$.) We then initialize a bisection scheme and locate the discontinuity point based on monitoring the size of the defect over each bisection interval. After the discontinuity is located, for example, at t_d, the interpolant associated with the interval $[t_n, t_n + h]$ is extended to $[t_n, t_d]$. This extension is easily done with the representation (2.2) by replacing h by $t_d - t_n$, $z_n(t_n + h)$ by $z_n(t_d)$, $z_n'(t_n + h)$ by $z_n'(t_d)$, and k_l by $z_n'(t_n + c_l(t_d - t_n))$, for $l = s+2, \ldots, \bar{s}+1$. This is the main reason to use the representation (2.2) rather than the representation (2.1) for the polynomial interpolant for evaluating the delay term. This strategy was found to be very effective for discontinuous problems for SDEs in (Enright et al. 1988). We only changed some parameters for this strategy and applied it to DDEs. From the numerical experiments we carried out, it seems to work effectively for a large class of retarded and neutral type DDEs.

A discontinuous problem for NDEs is more critical. As the order of derivative discontinuities will not smooth for NDEs, there may exist many low order derivative discontinuities propagated from the initial interval. The difficulty will be much severe if the delay vanishes in the integration interval. In such a situation, a cluster of low order derivative discontinuities occurs near the vanishing point. Although no method can locate all the discontinuity points for such problem, we expect that our technique may skip several discontinuity points at one time because the size of the errors arising from these discontinuities in such a small interval may be small. This prediction will fail for a class of problems (which we do not consider in this thesis) for which the size of the jumps at the discontinuity points is large.

2.3 An Iterative Scheme for Small/Vanishing Delays

The consideration in this section is the construction of an approximation $z(t)$ to evaluate the delay term in the current step. An approach for RDEs is first discussed in detail and an extension to NDEs is briefly described.

Vanishing delays and small delays cause a difficulty for numerical computation of DDEs. If a delay for the ith stage of a CRK formula is less than $c_i h_n$, then the evaluation of k_i (2.1) requires the evaluation of $z(t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i))$, which is not available. See Figure 2.1 below.

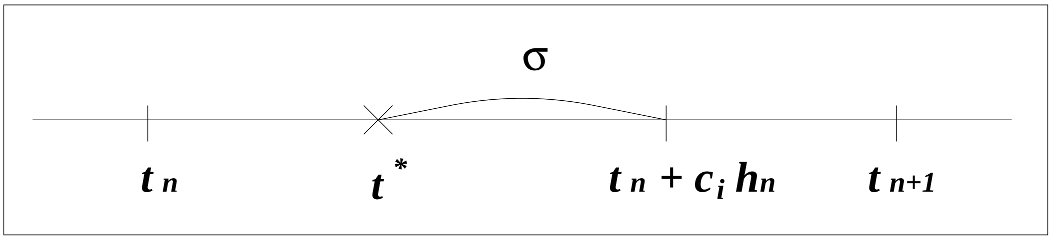

Figure 2.1: A Small Delay ($t^* = t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i)$)

An approach we develop for this problem is to first construct an initial approximation over the current step and then use a CRK formula iteratively. The iterative scheme (presented in

Algorithm 3) is used if the delay arguments associated with at least one stage is in the current step.

The initial approximation is either determined by extrapolation from the previous step $[t_{n-1}, t_n]$ (presented in Algorithm 1) or by a (low order) special interpolant over the current step (presented in Algorithm 2) depending on the state of the current step. In most steps, extrapolation from the previous step is used to evaluate the delay term for the initial approximation. More explicitly, if the step is the first step of integration or is to the right of a discontinuity (which coincided with the mesh point by the technique described in §2.2), extrapolation is not available or unreliable respectively. In these cases, a special interpolant is used to evaluate the required delay term. If some delay arguments are in the current step and the others are not in the current step, the initial approximation is used for the former and the standard interpolant is used for the latter.

Algorithm 1 (Initial Approximation : Extrapolation)
Given an initial value $y_n = z_{n-1}(t_n)$ and the piecewise polynomial $z(t)$ for $t \leq t_n$, this algorithm computes an initial approximation $z_n^{(0)}(t)$ for $t_n \leq t \leq t_{n+1}$ using extrapolation.

- 1st stage:

  $Y_1 = z_{n-1}(t_n), \qquad k_1 = f(t_n, z_{n-1}(t_n), z(t_n - \sigma(t_n, Y_1)))$

- ith stage: $i = 2$ to $\bar{s} + 1$

  $$Y_i^{(0)} = z_{n-1}(t_n) + h_n a_{i1} k_1 + h_n \sum_{j=2}^{i-1} a_{ij} k_j^{(0)}$$

  if $t_n < t_n + c_i h_n - \sigma(t_n, Y_i^{(0)}) \leq t_{n+1}$ **then**

  $\quad k_i^{(0)} = f(t_n + c_i h_n, Y_i^{(0)}, z_{n-1}(t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i^{(0)})))^\dagger$

  else

  $\quad k_i^{(0)} = f(t_n + c_i h_n, Y_i^{(0)}, z(t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i^{(0)})))$

  end if

- Calculate: $z_n^{(0)}(t_n + \tau h_n) = z_{n-1}(t_n) + h_n \bar{b}_1(\tau) k_1 + h_n \sum_{i=2}^{\bar{s}+1} \bar{b}_i(\tau) k_i^{(0)}$

† The interpolant $z_{n-1}(t)$ is evaluated outside the interval $[t_{n-1}, t_n]$. This corresponds to the case $\tau > 1$ in the representation (2.2).

Algorithm 2 (Initial Approximation : Special Interpolant)

Given an initial value $y_n = z_{n-1}(t_n)$ and the piecewise polynomial $z(t)$ for $t \leq t_n$, this algorithm computes an initial approximation $z_n^{(0)}(t)$ for $t_n \leq t \leq t_{n+1}$ using a special interpolant.

- 1st stage:

  $Y_1 = z_{n-1}(t_n), \qquad k_1 = f(t_n, z_{n-1}(t_n), z(t_n - \sigma(t_n, Y_1)))$

- 2nd stage:

  $Y_2^{(0)} = z_{n-1}(t_n) + h_n a_{21} k_1$

  if $t_n < t_n + c_2 h_n - \sigma(t_n, Y_2^{(0)}) \leq t_{n+1}$ **then**

  Construct a linear interpolant $z_{n2}^{(0)}(t)^{\dagger}$ defined by $z_{n-1}(t_n), Y_2^{(0)}$

  $k_2^{(0)} = f(t_n + c_2 h_n, Y_2^{(0)}, z_{n2}^{(0)}(t_n + c_2 h_n - \sigma(t_n + c_2 h_n, Y_2^{(0)})))$

  else

  $k_2^{(0)} = f(t_n + c_2 h_n, Y_2^{(0)} z(t_n + c_2 h_n - \sigma(t_n + c_2 h_n, Y_2^{(0)})))$

  end if

- ith stage: $i = 3$ to $\bar{s} + 1$

  $Y_i^{(0)} = z_{n-1}(t_n) + h_n a_{i1} k_1 + h_n \sum_{j=2}^{i-1} a_{ij} k_j^{(0)}$

  if $t_n < t_n + c_i h_n - \sigma(t_n, Y_i^{(0)}) \leq t_{n+1}$ **then**

  Construct a quadratic interpolant $z_{ni}^{(0)}(t)^{\ddagger}$ defined by $z_{n-1}(t_n), k_1, Y_i^{(0)}$

  $k_i^{(0)} = f(t_n + c_i h_n, Y_i^{(0)}, z_{ni}^{(0)}(t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i^{(0)})))$

  else

  $k_i^{(0)} = f(t_n + c_i h_n, Y_i^{(0)}, z(t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i^{(0)})))$

  end if

- Calculate: $z_n^{(0)}(t_n + \tau h_n) = z_{n-1}(t_n) + h_n \bar{b}_1(\tau) k_1 + h_n \sum_{i=2}^{\bar{s}+1} \bar{b}_i(\tau) k_i^{(0)}$

---

†This linear interpolant is determined uniquely by the points $[t_n, z_{n-1}(t_n)], [t_n + c_2 h_n, Y_2^{(0)}]$.

‡This quadratic interpolant is determined uniquely by the points $[t_n, z_{n-1}(t_n)], [t_n, k_1]$, $[t_n + c_i h_n, Y_i^{(0)}]$.

Algorithm 3 (Iteration Procedure)

Given an initial value $y_n = z_{n-1}(t_n)$, the piecewise polynomial $z(t)$ for $t \leq t_n$, the initial approximation $z_n^{(0)}(t)$ on the current step $[t_n, t_{n+1}]$, and k_1, this algorithm computes a continuous approximation $z_n^{(l)}(t)$ over $[t_n, t_{n+1}]$ iteratively, where $l = 1, 2, \cdots, m$ and m is the number of iterations. Then we define the approximation $z_n(t)$ by the mth approximation $z_n^{(m)}(t)$.

- ith stage: $i = 2$ to $\bar{s} + 1$

  $$Y_i^{(l)} = z_{n-1}(t_n) + h_n a_{i1} k_1 + h_n \sum_{j=2}^{i-1} a_{ij} k_j^{(l)},$$

  if $t_n < t_n + c_i h_n - \sigma(t_n, Y_i^{(0)}) \leq t_{n+1}$ **then**

  $$k_i^{(l)} = f(t_n + c_i h_n, Y_i^{(l)}, z_n^{(l-1)}(t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i^{(l)}))),$$

  else

  $$k_i^{(l)} = f(t_n + c_i h_n, Y_i^{(l)}, z(t_n + c_i h_n - \sigma(t_n + c_i h_n, Y_i^{(l)}))),$$

  end if

- Calculate: $z_n^{(l)}(t_n + \tau h_n) = z_{n-1}(t_n) + h_n \bar{b}_1(\tau) k_1 + h_n \sum_{i=2}^{\bar{s}+1} \bar{b}_i(\tau) k_i^{(l)}$

The initial approximation produced by extrapolation is a high order approximation to the delay value. In this case the local error $\|z_n^{(0)}(t_n + \tau h_n) - y_n(t_n + \tau h_n)\|$ will be proved to be $O(h_n^{p+1})$ in the next chapter, where p is the order of the underlying CRK formula. On the other hand, the associated error coefficient is usually large and we carry out a few iterations to compensate for this.

The order of the error of each special interpolant to the local solution is the same as the order of the local accuracy[2] of the corresponding stage. The local error $\|z_n^{(0)}(t_n + \tau h_n) - y_n(t_n + \tau h_n)\|$ in this case will be shown to be $O(h_n^{q+1})$ in the next chapter, where q is the minimum order of the local accuracy among all stages. The order of the interpolant $z_n^{(l)}$ will increase by one (up to the order of the underlying CRK formula) in each iteration. Thus, the number m of iterations per step for Algorithm 3 (necessary to achieve the order of the underlying CRK formula) is $p - q$.

For the class of continuous formulas we are considering, the derivative at t_{n+1} is

$$k_{s+1} = f(t_{n+1}, Y_{s+1}^{(m)}, z_n^{(m-1)}(t_{n+1} - \sigma(t_{n+1}, Y_{s+1}^{(m)}))),$$

[2]The local accuracy of the ith stage is defined by $\|y_n(t_n + c_i h_n) - Y_i\|$, where $y_n(t)$ is the local solution.

on the step $[t_n, t_{n+1}]$. However, on the next step $[t_{n+1}, t_{n+2}]$, the derivative at t_{n+1} is

$$k_1 = f(t_{n+1}, Y_{s+1}^{(m)}, z_n^{(m)}(t_{n+1} - \sigma(t_{n+1}, Y_{s+1}^{(m)}))).$$

Hence, the piecewise polynomial $z(t)$ may not be a C^1 interpolant anymore. To make the polynomial 'close' to a C^1 interpolant, we carry out one extra iteration on stage $s+1$ if some delay arguments for the internal stages between stage 2 and $s+1$ fall in the current step. This extra procedure is defined as follows.

$$\begin{aligned} k_{s+1}^{(m+1)} &= f(t_{n+1}, Y_{s+1}^{(m)}, z_n^{(m)}(t_{n+1} - \sigma(t_{n+1}, Y_{s+1}^{(m)}))), \\ z_n^{(m+1)}&(t_n + \tau h_n) \\ &= z_{n-1}(t_n) + h_n \bar{b}_1(\tau) k_1 + h_n \sum_{i=2}^{s} \bar{b}_i(\tau) k_i^{(m)} + h_n \bar{b}_{s+1}(\tau) k_{s+1}^{(m+1)} + h_n \sum_{i=s+2}^{\bar{s}+1} \bar{b}_i(\tau) k_i^{(m)}. \end{aligned}$$

Then we define the approximation $z_n(t)$ by the $(m+1)$st approximation $z_n^{(m+1)}(t_n + \tau h_n)$.

An extension of the iterative scheme to NDEs is made by replacing the initial approximation by its derivative and by replacing $z_n^{(l)}$ by its derivative $z_n'^{(l)}$ for $l = 1, 2, \cdots, m+1$. The local error $\|z_n^{(0)}(t_n + \tau h_n) - y_n(t_n + \tau h_n)\|$ associated with extrapolation will be shown to be $O(h_n^{p+1})$ in the next chapter but the local error associated with the special interpolant may only be $O(h_n^2)$. We will demonstrate in the next chapter that our method for NDEs works well in practice even though the theoretical order on such steps can be sub-optimal.

Chapter 3

Theoretical Results

In this chapter, we first establish sufficient conditions for convergence of the method described in Chapter 2. We then consider the order of the defect associated with the method. As our method tries to bound the defect by the prescribed tolerance, the order of the defect will reflect the potential cost of the approach. Finally, we show that the bound for global error associated with the method is proportional to the defect; therefore, the global error is bounded by a constant times tolerance.

In order to establish our convergence results, we assume that the solution is sufficiently smooth except for a finite number of points. We also assume that the function $f(t, y(t), z(t))$ is sufficiently smooth except for the points where $z(t)$ is not smooth. As the technique described §2.2 locates those discontinuity points and forces them to mesh points, this smoothness assumption is reasonable. Moreover, we assume that the left-hand and the right-hand rth derivative of the continuous approximation, $z(t)$, on mesh points (including discontinuous points) agree at least up to $O(h^{p-r+1})$ so that the theory of SDEs is valid, where p is the order of underlying CRK formula.

We make use of the following two lemmas in this chapter. The first lemma is the key step for a proof of convergence. This lemma was used to prove the order of convergence of another method for RDEs in (Baker and Paul 1993b).

Lemma 1 *If the sequence $\{e_n\}$ satisfies the difference inequality,*

$$e_n \leq (1 + Lh_n)e_{n-1} + d_n h_n, \quad \textit{for} \quad n = 0, 1, \ldots,$$

where $\{e_n\}, \{d_n\}$, *and* $\{h_n\}$ *are nonnegative sequences, and* L *is a nonnegative constant, then*

$$e_n \leq \left(e_{-1} + \sum_{i=0}^{n} d_i h_i\right) \exp\left(\sum_{j=0}^{n} h_j L\right).$$

The following Gronwall inequality (Brauer and Nohel 1989, p. 32) is applied in §3.1.3 and §3.2.3 when we derive global error bounds.

Lemma 2 *Let* K_1 *and* K_2 *be nonnegative constants and let* u *be a continuous nonnegative function on an interval* $a \leq t \leq b$ *satisfying the inequality*

$$u(t) \leq K_1 + K_2 \int_a^t u(s)\, ds, \quad a \leq t \leq b,$$

then

$$u(t) \leq K_1 \exp[K_2(t-a)], \quad a \leq t \leq b.$$

3.1 Numerical Solution of RDEs

In this section, we shall use the inequality

$$\begin{aligned} &\|z(t-\sigma(t,z(t))) - y(t-\sigma(t,y(t)))\| \\ &\quad \leq \quad \|z(t-\sigma(t,z(t))) - y(t-\sigma(t,z(t)))\| + L_y L_\sigma \|z(t) - y(t)\|, \end{aligned} \tag{3.1}$$

for the delay term associated with state dependent delay problems when the functions y and σ satisfy the Lipschitz conditions (1.8) and (1.10). The inequality (3.1) is proved by applying the triangle inequality along with the Lipschitz conditions as follows.

$$\begin{aligned} &\|z(t-\sigma(t,z(t))) - y(t-\sigma(t,y(t)))\| \\ &\quad \leq \quad \|z(t-\sigma(t,z(t))) - y(t-\sigma(t,z(t)))\| + \|y(t-\sigma(t,z(t))) - y(t-\sigma(t,y(t)))\| \\ &\quad \leq \quad \|z(t-\sigma(t,z(t))) - y(t-\sigma(t,z(t)))\| + L_y \|\sigma(t,z(t)) - \sigma(t,y(t))\| \\ &\quad \leq \quad \|z(t-\sigma(t,z(t))) - y(t-\sigma(t,z(t)))\| + L_y L_\sigma \|z(t) - y(t)\|. \end{aligned}$$

3.1.1 Convergence

In this subsection we identify conditions which will imply that a method is convergent. Convergence (which is defined in §1.2.1) is necessary for a numerical method to be practical.

Theorem 1 *If the functions f, y, σ satisfy the Lipschitz conditions (1.7),(1.8),(1.10) and the method (2.1) is consistent, then the method is convergent.*

Proof. We first consider time dependent/constant delay problems.

On expanding the solution of (1.4) about t_n in a Taylor series, we have

$$y(t_n + \tau h_n) = y(t_n) + \tau h_n f(t_n, y(t_n), y(t_n - \sigma)) + O(h_n^2), \tag{3.2}$$

for $0 \le \tau \le 1$. Subtracting (3.2) from the consistency condition (1.13) and defining

$$e_n = \max_{t_0 \le t \le t_{n+1}} \|z(t) - y(t)\|,$$

we obtain

$$\begin{aligned} \|z_n(t_n + \tau h_n) - y(t_n + \tau h_n)\| &\le e_{n-1} + \tau h_n (L_1 + L_2) e_{n-1} + O(h_n^2) \\ &\le [1 + h_n(L_1 + L_2)] e_{n-1} + C h_n^2, \end{aligned}$$

where C is a nonnegative constant. This inequality is satisfied for any $\tau \in [0, 1]$ and the right-hand side is greater than e_{n-1}; therefore,

$$e_n \le [1 + h_n(L_1 + L_2)] e_{n-1} + C h_n^2.$$

Applying Lemma 1 to the above inequality yields

$$\begin{aligned} e_n &\le \left(e_{-1} + \sum_{i=0}^{n} C h_i^2\right) \exp\left[\sum_{j=0}^{n} h_j (L_1 + L_2)\right] \\ &\le CH \sum_{i=0}^{n} h_i \exp\left[(L_1 + L_2) \sum_{j=0}^{n} h_j\right] \\ &\le TCH \exp[(L_1 + L_2)T], \end{aligned}$$

where $H = \max_i h_i$ and $T = \sum h_i$. (In the above inequalities, we used the fact that the initial error, e_{-1}, is 0.) It follows that $e_n \to 0$ as $H \to 0$. Hence, convergence is established.

The analysis for state dependent delay problems now follows. The solution of (1.4) for the state dependent delay case satisfies

$$y(t_n + \tau h_n) = y(t_n) + \tau h_n f(t_n, y(t_n), y(t_n - \sigma(t_n, y(t_n)))) + O(h_n^2). \tag{3.3}$$

Subtracting (3.3) from the consistency condition (1.13) and defining

$$e_n = \max_{t_0 \le t \le t_{n+1}} \|z(t) - y(t)\|,$$

we obtain

$$\begin{aligned} &\|z_n(t_n + \tau h_n) - y(t_n + \tau h_n)\| \\ &\quad \le e_{n-1} + \tau h_n [L_1 e_{n-1} + L_2 \|z(t_n - \sigma(t_n, z(t_n))) - y(t_n - \sigma(t_n, y(t_n)))\|] + O(h_n^2). \end{aligned}$$

Substituting (3.1) into the above inequality leads to

$$\begin{aligned} e_n &\le (1 + h_n L_1) e_{n-1} + h_n L_2 [\|z(t_n - \sigma(t_n, z(t_n))) - y(t_n - \sigma(t_n, z(t_n)))\| \\ &\quad + L_y L_\sigma \|z(t_n) - y(t_n)\|] + C h_n^2 \\ &\le [1 + h_n (L_1 + L_2 + L_2 L_y L_\sigma)] e_{n-1} + C h_n^2, \end{aligned}$$

where C is a nonnegative constant.

Applying Lemma 1 to the above inequality yields

$$e_n \le TCH \exp[(L_1 + L_2 + L_2 L_y L_\sigma) T].$$

Thus, $e_n \to 0$ as $H \to 0$ and hence convergence follows for the state dependent delay case.

$\square$

3.1.2 Order of the Defect

In order to determine the order of the defect we must determine the order of the method. To this end, we first prove a lemma which identifies the order of the method. Then this lemma is applied to three different cases and the order of the method in each case is derived. Finally, the order of the defect is identified.

Let us suppose that a CRK formula

$$\begin{aligned}
\tilde{z}_n(t_n+\tau h_n) &= z_{n-1}(t_n) + h_n \sum_{i=1}^{\bar{s}+1} \bar{b}_i(\tau)\tilde{k}_i, \\
\tilde{k}_i &= f(t_n + c_i h_n, \tilde{Y}_i, y_n(t_n + c_i h_n - \sigma(t_n + c_i h_n, \tilde{Y}_i))), \\
\tilde{Y}_i &= z_{n-1}(t_n) + h_n \sum_{j=1}^{i-1} a_{ij}\tilde{k}_j,
\end{aligned} \tag{3.4}$$

with the exact local solution for computing the delay term is applied to the local problem (1.11). Then the local error associated with $\tilde{z}_n(t_n+\tau h_n)$ in approximating the local solution $y_n(t_n+\tau h_n)$ is $O(h_n^{p+1})$, where p is the order of the underlying CRK formula when applied to SDEs because the delay term is exactly determined by the local solution[1]. In the following lemma, we assume that the delay term is approximated by a function $z(t)$ such that $\|z(t) - y_n(t)\| = O(h_n^q)$ for $t \leq t_n + h_n$ instead of using the exact local solution. Then we compare the numerical solution with $\tilde{z}_n(t)$ and identify the order of the method.

Lemma 3 *If the following conditions are satisfied,*

1. *a pth order CRK formula is applied to the problem (1.11) with an approximation $z(t)$ to $y_n(t)$ for computing the delay term,*
2. *the error of $z(t)$ to approximate $y_n(t)$ is $O(h_n^q)$,*
3. *the functions f, y_n, σ satisfy the Lipschitz conditions (1.7),(1.8),(1.10),*

then the order of the method is r, where $r = \min(p,q)$.

Proof. A CRK formula associated with $z(t)$ is defined by (2.1). We shall first show that $\|k_i - \tilde{k}_i\| = O(h_n^q)$ by induction on i.
Since the formula we are considering is an explicit RK formula, for $i = 1$, $Y_1 = \tilde{Y}_1 = z(t_n)$; therefore, it follows that

$$\|k_1 - \tilde{k}_1\| = \|f(t_n, Y_1, z(t_n - \sigma(t_n, Y_1))) - f(t_n, \tilde{Y}_1, y_n(t_n - \sigma(t_n, \tilde{Y}_1)))\|$$

[1]Although the local solution depends on the continuous approximation, $z(t)$, this statement is valid since we assumed that the left- and the right-hand rth derivative of the continuous approximation, $z(t)$, agree at least up to $O(h^{p-r+1})$ on mesh points.

$$
\begin{aligned}
&\leq L_2\|z(t_n-\sigma(t_n,Y_1))-y_n(t_n-\sigma(t_n,Y_1))\| \\
&= O(h_n^q).
\end{aligned}
$$

Now assume that $\|k_i-\tilde{k}_i\|=O(h_n^q)$ for $i\leq m-1<\bar{s}+1$. Then

$$\|Y_m-\tilde{Y}_m\|\leq h_n\sum_{j=1}^{m-1}|a_{mj}|\|k_j-\tilde{k}_j\|=O(h_n^{q+1}),$$

since the quantity $\sum_{j=1}^{m-1}|a_{mj}|$ is uniformly bounded. Using the above inequality and the Lipschitz condition (1.7), we obtain

$$
\begin{aligned}
\|k_m-\tilde{k}_m\| &= \|f(t_n+c_mh_n,Y_m,z(t_n+c_mh_n-\sigma(t_n+c_mh_n,Y_m))) \\
&\quad - f(t_n+c_mh_n,\tilde{Y}_m,y_n(t_n+c_mh_n-\sigma(t_n+c_mh_n,\tilde{Y}_m)))\| \\
&\leq L_1\|Y_m-\tilde{Y}_m\|+L_2\|z(t_n+c_mh_n-\sigma(t_n+c_mh_n,Y_m)) \\
&\quad - y_n(t_n+c_mh_n-\sigma(t_n+c_mh_n,\tilde{Y}_m))\|.
\end{aligned}
$$

Substitution of (3.1) into the above inequality leads to

$$
\begin{aligned}
\|k_m-\tilde{k}_m\| &\leq L_1\|Y_m-\tilde{Y}_m\|+L_2[\|z(t_n+c_mh_n-\sigma(t_n+c_mh_n,Y_m)) \\
&\quad - y_n(t_n+c_mh_n-\sigma(t_n+c_mh_n,Y_m))\|+L_yL_\sigma\|Y_m-\tilde{Y}_m\|] \\
&= L_1O(h_n^{q+1})+L_2[O(h_n^q)+L_yL_\sigma O(h_n^{q+1})] \\
&= O(h_n^q).
\end{aligned}
$$

It follows that the induction hypothesis is true for $i\leq m$ and that

$$\|k_i-\tilde{k}_i\|=O(h_n^q), \quad \text{for} \quad i=1,2,\cdots,\bar{s}+1.$$

Consequently,

$$\|z_n(t_n+\tau h_n)-\tilde{z}_n(t_n+\tau h_n)\| \;\leq\; h_n\sum_{i=1}^{\bar{s}+1}|b_i(\tau)|\;\|k_i-\tilde{k}_i\| \;=\; O(h_n^{q+1}).$$

Applying the triangle inequality and using the above result that $\|z_n(t)-\tilde{z}_n(t)\|=O(h_n^{q+1})$, we conclude that

$$\begin{aligned}\|z_n(t) - y_n(t)\| &\leq \|z_n(t) - \tilde{z}_n(t)\| + \|\tilde{z}_n(t) - y_n(t)\| \\ &= O(h_n^{q+1}) + O(h_n^{p+1}) \\ &= O(h_n^{\min(p+1,q+1)}).\end{aligned}$$

□

We now study the order of the method for three separate cases by applying Lemma 3.

Case 1: No delays in the current step.

Here $z(t) = y_n(t)$ by the definition of the local problem (1.11). Hence, by Lemma 3, the order of the method is p.

Case 2: Some delays in the current step computed with the special interpolant for the initial approximation.

The order of the local accuracy[2] of the CRK formula we use is 2 for the second stage and 3 for the others. The local accuracy will not be changed by the presence of the delay term. Note that this is not true for NDEs, which will be discussed in §3.2.2. From Lemma 3, the order of the method is 2 after the initial approximation. However, the $O(h_n^2)$ term arising in the error for k_2 does not affect the order of the method since the $O(h_n^2)$ term associated with k_2 will contribute to the $O(h_n^3)$ term of Y_3 and $b_2(\tau) = 0$ for the formula we use. In other words, the order of the method is 3 after the initial approximation. Thus, we need $p - 3$ more iterations to achieve pth order.

Case 3: Some delays in the current step computed with extrapolation for the initial approximation.

The order of the extrapolation from the previous step, $[t_{n-1}, t_n]$, only depends on the degree of the interpolating polynomial in the current step, $[t_n, t_{n+1}]$, because the local problem (1.11) is defined such that

$$y_n(t) = z(t), \quad \text{for} \quad t \leq t_n.$$

We assume that the degree of the polynomial is p and that the current stepsize h_n is small

[2] For the definition of the local accuracy see p. 21.

enough compared to the previous stepsize h_{n-1}. Then we have

$$\|z(t) - y_n(t)\| \le Ch_{n-1}^{p+1} \quad \text{on} \quad t_{n-1} \le t \le t_n.$$

In the current step, we have

$$\|z(t) - y_n(t)\| \le C((1+\gamma)h_{n-1})^{p+1} \le C\left(\frac{1+\gamma}{\gamma}h_n\right)^{p+1} \le C' h_n^{p+1},$$

where $\gamma h_{n-1} = h_n$ and $C' = C\left(\dfrac{1+\gamma}{\gamma}\right)^{p+1}$. We note that γ cannot approach to 0 (except at discontinuous points) because of smoothness assumption. Therefore, from Lemma 3, it follows that the order of the method is p.

We establish the order of the defect in the following theorem.

Theorem 2 *If the order of the method for RDEs is p and the functions f, y, σ satisfy the Lipschitz conditions (1.7),(1.8),(1.10), then $\|\delta_n(t)\| = O(h_n^p)$.*

Proof. Applying the triangle inequality and using the Lipschitz condition (1.7), we find

$$\begin{aligned}
\|\delta_n(t)\| &= \|z_n'(t) - f(t, z_n(t), z(t - \sigma(t, z_n(t))))\| \\
&\le \|z_n'(t) - y_n'(t)\| + \|f(t, z_n(t), z(t - \sigma(t, z_n(t)))) - f(t, y_n(t), y_n(t - \sigma(t, y_n(t))))\| \\
&\le \|z_n'(t) - y_n'(t)\| + L_1\|z_n(t) - y_n(t)\| + L_2\|z(t - \sigma(t, z_n(t))) - y_n(t - \sigma(t, y_n(t)))\|.
\end{aligned}$$

Substituting (3.1) into the above inequality results in

$$\begin{aligned}
&\|\delta_n(t)\| \le \|z_n'(t) - y_n'(t)\| + L_1\|z_n(t) - y_n(t)\| \\
&\quad + L_2[\|z(t - \sigma(t, z_n(t))) - y_n(t - \sigma(t, z_n(t)))\| + L_y L_\sigma \|z_n(t) - y_n(t)\|].
\end{aligned}$$

Because the order of the method is p, we obtain the desired result

$$\|\delta_n(t)\| \le O(h_n^p) + L_1 O(h_n^{p+1}) + L_2[O(h^{p+1}) + L_y L_\sigma O(h_n^{p+1})] = O(h_n^p).$$

□

We note that since $z_n(t)$ is a polynomial and sufficient smoothness of the solution is assumed, the difference, $z_n(t) - y_n(t)$, can be expanded in a Taylor series as

$$z_n(t_n + \tau h_n) - y_n(t_n + \tau h_n) = q_{p+1}(\tau) h_n^{p+1} + O(h_n^{p+2}) = \sum_{j \geq 1} q_{p+j}(\tau) h_n^{p+j},$$

except for a finite number of isolated points. Hence, the derivative of this difference with respect to t can be expressed as

$$\begin{aligned} z_n'(t_n + \tau h_n) - y_n'(t_n + \tau h_n) &= \frac{1}{h_n} \sum_{j \geq 1} q_{p+j}'(\tau) h_n^{p+j} = \sum_{j \geq 1} q_{p+j}'(\tau) h_n^{p+j-1} \\ &= q_{p+1}'(\tau) h_n^p + O(h_n^{p+1}). \end{aligned}$$

For the finite number of isolated points where the delay argument is on some previous mesh point, because of the continuity assumption, the result still holds.

3.1.3 A Global Error Bound

We now show that the bound for global error associated with the method is proportional to the defect.

Theorem 3 *If the norm of the defect, $\|\delta(t)\|$, is bounded by TOL on $[t_0, t_F]$, and the functions f, y, σ satisfy the Lipschitz conditions (1.7),(1.8),(1.10), then the global error, $\|z(t) - y(t)\|$, is bounded by $C \cdot TOL$ on $[t_0, t_F]$, where C is a nonnegative constant.*

Proof. The solution for the RDE (1.4) satisfies the differential equation

$$y'(t) = f(t, y(t), y(t - \sigma(t, y(t)))). \tag{3.5}$$

The numerical solution satisfies the perturbed differential equation

$$z'(t) = f(t, z(t), z(t - \sigma(t, z(t)))) + \delta(t), \tag{3.6}$$

on $[t_0, t_F]$.

Subtracting (3.5) from (3.6), integrating, and taking a norm of both sides, we obtain

$$\|z(t) - y(t)\| \leq \int_{t_0}^{t} [L_1 \|z(x) - y(x)\| + L_2 \|z(x - \sigma(t, z(x))) - y(x - \sigma(t, y(x)))\| + \|\delta(x)\|] \, dx.$$

Let $\alpha(t) = \|z(t) - y(t)\|$ and $\beta(t) = \max\limits_{t_0 \le s \le t} \alpha(s)$. Substituting α and β, and using $\|\delta(t)\| \le \mathrm{TOL}$ lead to

$$\alpha(t) \le \int_{t_0}^{t} [L_1\beta(x) + L_2\|z(x - \sigma(t, z(x))) - y(x - \sigma(t, y(x)))\|]\, dx + (t_F - t_0)\mathrm{TOL}.$$

Further, the substitution of the inequality (3.1) into the above inequality implies

$$\begin{aligned} \alpha(t) &\le \int_{t_0}^{t} [L_1\beta(x) + L_2\|z(x - \sigma(x, z(x))) - y(x - \sigma(x, z(x)))\| \\ &\quad + L_2 L_y L_\sigma \|z(x) - y(x)\|]\, dx + (t_F - t_0)\mathrm{TOL} \\ &\le (L_1 + L_2 + L_2 L_y L_\sigma) \int_{t_0}^{t} \beta(x) dx + (t_F - t_0)\mathrm{TOL}. \end{aligned}$$

Because the right-hand side of the above inequality is monotone increasing, we find

$$\begin{aligned} \alpha(t) \le \beta(t) = \max_{t_0 \le s \le t} \alpha(s) = \alpha(s^*) &\le (L_1 + L_2 + L_2 L_y L_\sigma) \int_{t_0}^{s^*} \beta(x) dx + (t_F - t_0)\mathrm{TOL} \\ &\le (L_1 + L_2 + L_2 L_y L_\sigma) \int_{t_0}^{t} \beta(x) dx + (t_F - t_0)\mathrm{TOL}, \end{aligned}$$

where $t_0 \le s^* \le t$.

Applying Lemma 2 to the above inequality yields

$$\beta(t) \le (t_F - t_0)\mathrm{TOL} \exp[(L_1 + L_2 + L_2 L_y L_\sigma)(t_F - t_0)].$$

We hence can conclude that

$$\|z(t) - y(t)\| \le (t_F - t_0)\mathrm{TOL} \exp[(L_1 + L_2 + L_2 L_y L_\sigma)(t_F - t_0)] = C \cdot \mathrm{TOL},$$

where $C = (t_F - t_0) \exp[(L_1 + L_2 + L_2 L_y L_\sigma)(t_F - t_0)]$.

$\square$

Remark: It follows from Theorem 3 that the global error is bounded by a multiple of tolerance as long as the defect is bounded by tolerance regardless of the order of the defect. This becomes significant when we consider NDEs in §3.2.3 where the order of the defect may be less than p on some steps.

3.2 Numerical Solution of NDEs

In this section some results for RDEs from §3.1 are extended to the case of NDEs. We present an analysis for state dependent problems only.

Throughout this section, we shall use the inequality (analogous to (3.1))

$$\begin{aligned} &\|z'(t-\sigma(t,z(t))) - y'(t-\sigma(t,y(t)))\| \\ &\quad \leq \|z'(t-\sigma(t,z(t))) - y'(t-\sigma(t,z(t)))\| + L_{y'}L_\sigma\|z(t)-y(t)\|, \end{aligned} \tag{3.7}$$

for the derivative delay term associated with state dependent problems when the functions y' and σ satisfy the Lipschitz conditions (1.9) and (1.10). This inequality can be proved by applying the triangle inequality along with the Lipschitz conditions.

Furthermore, in this section, we shall make use of the assumption that the Lipschitz constant L_3 satisfies

$$0 < L_3 < 1.$$

Recall that this condition is one of the sufficient conditions for the existence and uniqueness of solutions to an NDE. Note that for the multiple delay case, we require the sum of the Lipschitz constants with respect to derivative delay terms must satisfies the condition

$$0 < \sum_i |L_3^{(i)}| < 1,$$

to establish the results presented in this section.

3.2.1 Convergence

Theorem 4 *If the functions f, y', σ satisfy the Lipschitz conditions (1.7),(1.9),(1.10) and the method (2.3) is consistent, then the method is convergent.*

Proof. On expanding the solution of (1.5) about t_n in a Taylor series, we have

$$y(t_n + \tau h_n) = y(t_n) + \tau h_n f(t_n, y(t_n), y'(t_n - \sigma(t_n, y(t_n)))) + O(h_n^2). \tag{3.8}$$

Subtracting (3.8) from the consistency condition (1.14) and defining

$$e_n = \max_{t_0 \le t \le t_{n+1}} \|z(t) - y(t)\|,$$

we have

$$\begin{aligned} &\|z_n(t_n + \tau h_n) - y(t_n + \tau h_n)\| \\ &\quad \le \; e_{n-1} + \tau h_n[L_1 e_{n-1} + L_3 \|z'(t_n - \sigma(t_n, z(t_n))) - y'(t_n - \sigma(t_n, y(t_n)))\|] + O(h_n^2). \end{aligned}$$

Substituting (3.7) into the above inequality and defining

$$e'_n = \max_{t_0 \le t \le t_{n+1}} \|z'(t) - y'(t)\|,$$

we obtain

$$\begin{aligned} \|z_n(t_n + \tau h_n) - y(t_n + \tau h_n)\| \; &\le \; e_{n-1} + h_n[L_1 e_{n-1} + L_3(e'_{n-1} + L_{y'} L_\sigma e_{n-1})] + C_1 h_n^2 \\ &\le \; [1 + h_n(L_1 + L_3 L_{y'} L_\sigma)]e_{n-1} + h_n L_3 e'_{n-1} + C_1 h_n^2, \end{aligned}$$

where C_1 is a nonnegative constant. The above inequality is satisfied for any $\tau \in [0, 1]$ and the right-hand side is greater than e_{n-1}. Hence,

$$e_n \le [1 + h_n(L_1 + L_3 L_{y'} L_\sigma)]e_{n-1} + h_n L_3 e'_{n-1} + C_1 h_n^2. \tag{3.9}$$

Subtracting the derivative of (3.8) with respect to t from the derivative of (1.14)[3] and using the inequality (3.7), we find

$$\begin{aligned} &\|z'_n(t_n + \tau h_n) - y'(t_n + \tau h_n)\| \\ &\quad \le \; L_1 e_{n-1} + L_3 \|z'(t_n - \sigma(t_n, z(t_n))) - y'(t_n - \sigma(t_n, y(t_n)))\| + O(h_n), \\ &\quad \le \; (L_1 + L_3 L_{y'} L_\sigma)e_{n-1} + L_3 e'_{n-1} + C_2 h_n, \end{aligned}$$

where C_2 is a nonnegative constant which, because of our smoothness assumption, can be as-

[3]For the justification of this approach, see the note on pp. 30–31.

sumed to be independent of n. Using the definition of e_n and e'_n leads to

$$\|z'_n(t_n + \tau h_n) - y'(t_n + \tau h_n)\| \leq (L_1 + L_3 L_{y'} L_\sigma) e_n + L_3 e'_n + C_2 h_n.$$

Suppose that $e'_n = \|z'_i(t_i + \tau^* h_i) - y'(t_i + \tau^* h_i)\|$ for $0 \leq i \leq n$ and $\tau^* \in [0,1]$. Then from the above inequality, we find

$$\begin{aligned} e'_n = \|z'_i(t_i + \tau^* h_i) - y'(t_i + \tau^* h_i)\| &\leq (L_1 + L_3 L_{y'} L_\sigma) e_i + L_3 e'_i + C_2 h_i \\ &\leq (L_1 + L_3 L_{y'} L_\sigma) e_n + L_3 e'_n + C_2 H, \end{aligned}$$

where $H = \max_i h_i$. As $0 < L_3 < 1$, the above inequality becomes

$$e'_n \leq \frac{1}{1 - L_3}(L_1 + L_3 L_{y'} L_\sigma) e_n + \frac{1}{1 - L_3} C_2 H. \tag{3.10}$$

Letting $L = L_1 + L_3 L_{y'} L_\sigma$ and substituting (3.10) into (3.9) results in

$$\begin{aligned} e_n &\leq (1 + h_n L) e_{n-1} + h_n L_3 \frac{L}{1 - L_3} e_{n-1} + h_n L_3 \frac{1}{1 - L_3} C_2 H + C_1 h_n^2 \\ &\leq \left(1 + \frac{L}{1 - L_3} h_n\right) e_{n-1} + \left(C_1 + \frac{C_2 L_3}{1 - L_3}\right) h_n H. \end{aligned}$$

Applying Lemma 1 to the above inequality yields

$$\begin{aligned} e_n &\leq \left[e_{-1} + \sum_{i=0}^{n} \left(C_1 + \frac{C_2 L_3}{1 - L_3}\right) h_i H\right] \exp\left(\sum_{j=0}^{n} h_j \frac{L}{1 - L_3}\right) \\ &\leq TH \left(C_1 + \frac{C_2 L_3}{1 - L_3}\right) \exp\left(\frac{LT}{1 - L_3}\right), \end{aligned}$$

where $T = \sum h_i$.

Hence $e_n \to 0$ as $H \to 0$. It follows that the numerical solution, $z(t)$, is convergent. Moreover, it follows from (3.10) that the derivative of the numerical solution, $z'(t)$, is also convergent.

$\square$

3.2.2 Order of the Defect

We begin by considering the order of the local error as in §3.1.2.

Lemma 4 *If the following conditions are satisfied,*

1. *a pth order CRK formula is applied to the problem (1.12) with an approximation $z'(t)$ to $y_n'(t)$ for computing the delay term,*

2. *the error of $z'(t)$ to approximate $y_n'(t)$ is $O(h_n^{q-1})$,*

3. *the functions f, y_n', σ satisfy the Lipschitz conditions (1.7),(1.9),(1.10),*

then the order of the method is r, where $r = \min(p, q-1)$.

The proof of this lemma is the same as that of Lemma 3 except that the order of the approximation to the delay term is decreased by one as is the order of the method.

We again study the order of the method for three separate cases by applying Lemma 4.

Case 1: No delays in the current step.

Here $z'(t) = y_n'(t)$ by the definition of the local problem (1.12). Hence, by Lemma 4, the order of the method is p.

Case 2: Some delays in the current step computed with the special interpolant for the initial approximation.

The order of the local accuracy of the CRK formula we use is 2 for the second stage and 3 for the others. The order of error in approximating the derivative delay term is one less than the corresponding order of local accuracy; thus,

$$\begin{aligned}
&\|z_{n2}'^{(0)}(t_n + c_2h_n - \sigma) - y_n'(t_n + c_2h_n - \sigma)\| = O(h_n),\\
&\|k_2^{(0)} - \tilde{k}_2\| \le L_1\|Y_2^{(0)} - \tilde{Y}_2\| + L_3\|z_{n2}'^{(0)}(t_n + c_2h_n - \sigma) - y_n'(t_n + c_2h_n - \sigma)\| = O(h_n),\\
&\|Y_3^{(0)} - \tilde{Y}_3\| \le h_n a_{32}\|k_2^{(0)} - \tilde{k}_2\| = O(h_n^2),\\
&\|z_{n3}'^{(0)}(t_n + c_3h_n - \sigma) - y_n'(t_n + c_3h_n - \sigma)\| = O(h_n),
\end{aligned}$$

which imply

$$\|z_{ni}'^{(0)}(t_n + c_ih_n - \sigma) - y_n'(t_n + c_ih_n - \sigma)\| = O(h_n), \quad \text{for} \quad i = 2, 3, \cdots, \bar{s}+1.$$

It follows from Lemma 4 that the order of the method is $r = \min(p, 1) = 1$. We remark that pth order cannot be achieved in this case even with the iteration procedure (Algorithm 3) because of the fact that

$$\|z_n'^{(0)}(t) - y_n'(t)\| = O(h_n), \quad \text{for} \quad t_n \le t \le t_{n+1}.$$

This implies (from Lemma 4) that the order of the method in the next iteration is still one. Note that this low order will not affect the order of the following steps since our definition of order is a local property.

Case 3: Some delays in the current step computed with extrapolation for the initial approximation.

The order of the extrapolation from the previous step, $[t_{n-1}, t_n]$, depends only on the degree of the interpolating polynomial in the current step, $[t_n, t_{n+1}]$. We assume that the degree of the polynomial is p and the current stepsize h_n is small compared to the previous stepsize h_{n-1}. Then, we have

$$\|z'(t) - y_n'(t)\| \leq Ch_{n-1}^p \quad \text{on} \quad t_{n-1} \leq t \leq t_n.$$

In the current step, we have

$$\|z'(t) - y_n'(t)\| \leq C((1+\gamma)h_{n-1})^p \leq C\left(\frac{1+\gamma}{\gamma}h_n\right)^p \leq C'h_n^p,$$

where $\gamma h_{n-1} = h_n$ and $C' = C\left(\frac{1+\gamma}{\gamma}\right)^p$. From Lemma 4, it follows that the order of the method is p.

We now study the order of the defect. Note that the order of the method q may be less than p.

Theorem 5 *If the order of the method for NDEs is q and the functions f, y', σ satisfy the Lipschitz conditions (1.7), (1.9),(1.10), then $\|\delta_n(t)\| = O(h_n^q)$.*

Proof. Applying the triangle inequality and using the Lipschitz condition (1.7), we have

$$\begin{aligned}
\|\delta_n(t)\| &= \|z_n'(t) - f(t, z_n(t), z'(t - \sigma(t, z_n(t))))\| \\
&\leq \|z_n'(t) - y_n'(t)\| + \|f(t, z_n(t), z'(t - \sigma(t, z_n(t)))) - f(t, y_n(t), y_n'(t - \sigma(t, y_n(t))))\| \\
&\leq \|z_n'(t) - y_n'(t)\| + L_1\|z_n(t) - y_n(t)\| + L_3\|z'(t - \sigma(t, z_n(t))) - y_n'(t - \sigma(t, y_n(t)))\|.
\end{aligned}$$

Substituting (3.7) into the above inequality leads to

$$\begin{aligned}
\|\delta_n(t)\| \leq\ & \|z_n'(t) - y_n'(t)\| + L_1\|z_n(t) - y_n(t)\| \\
& + L_3[\|z'(t - \sigma(t, z_n(t))) - y_n'(t - \sigma(t, z_n(t)))\| + L_{y'}L_\sigma\|z_n(t) - y_n(t)\|].
\end{aligned}$$

As the order of the method is q, we obtain the desired result

$$\|\delta_n(t)\| \leq O(h_n^q) + L_1 O(h_n^{q+1}) + L_3[O(h_n^q) + L_{y'} L_\sigma O(h_n^{q+1})] = O(h_n^q).$$

$\square$

3.2.3 A Global Error Bound

We show here that the global error of our numerical solution for NDEs is bounded on the whole interval by a multiple of tolerance.

Theorem 6 *If the norm of the defect, $\|\delta(t)\|$, is bounded by TOL on $[t_0, t_F]$, and the functions f, y', σ satisfy the Lipschitz conditions (1.7),(1.9),(1.10), then the global error, $\|z(t) - y(t)\|$, is bounded by $C \cdot TOL$ on $[t_0, t_F]$, where C is a nonnegative constant.*

Proof. The solution of the NDE (1.5) satisfies the differential equation

$$y'(t) = f(t, y(t), y'(t - \sigma(t, y(t)))), \tag{3.11}$$

and the numerical solution satisfies the perturbed differential equation

$$z'(t) = f(t, z(t), z'(t - \sigma(t, z(t)))) + \delta(t), \tag{3.12}$$

on $[t_0, t_F]$. Subtracting (3.11) from (3.12), integrating, and taking a norm of both sides, we obtain

$$\begin{aligned} \|z(t) - y(t)\| \quad \leq \quad & \int_{t_0}^{t} [L_1 \|z(x) - y(x)\| \\ & + L_3 \|z'(x - \sigma(x, z(x))) - y'(x - \sigma(x, y(x)))\| + \|\delta(x)\|] \, dx. \end{aligned}$$

Let $\alpha(t) = \|z(t) - y(t)\|, \beta(t) = \max_{t_0 \leq s \leq t} \alpha(s)$, and $\gamma(t) = \max_{t_0 \leq s \leq t} \|z'(s) - y'(s)\|$. Substituting α, β into the above inequality, and using $\|\delta(t)\| \leq$ TOL leads to

$$\alpha(t) \leq \int_{t_0}^{t} [L_1 \beta(x) + L_3 \|z'(x - \sigma(x, z(x))) - y'(x - \sigma(x, y(x)))\|] \, dx + (t_F - t_0)\mathrm{TOL}.$$

Further, substitution of (3.7) and γ into the above inequality results in

$$\begin{aligned}
\alpha(t) &\leq \int_{t_0}^{t}[L_1\beta(x) + L_3\|z'(x-\sigma(x,z(x))) - y'(x-\sigma(x,z(x)))\| + L_3L_{y'}L_\sigma\|z(x)-y(x)\|]\,dx \\
&\quad + (t_F - t_0)\mathrm{TOL} \\
&\leq \int_{t_0}^{t}[(L_1 + L_3L_{y'}L_\sigma)\beta(x) + L_3\gamma(x)]\,dx + (t_F - t_0)\mathrm{TOL}. \qquad (3.13)
\end{aligned}$$

Subtracting (3.11) from (3.12), and using (3.7), we obtain

$$\begin{aligned}
\|z'(t) - y'(t)\| &\leq \|f(t,z(t),z'(t-\sigma(t,z(t)))) - f(t,y(t),y'(t-\sigma(t,y(t))))\| + \|\delta(t)\| \\
&\leq (L_1 + L_3L_{y'}L_\sigma)\|z(t)-y(t)\| \\
&\quad + L_3\|z'(t-\sigma(t,z(t))) - y'(t-\sigma(t,z(t)))\| + \mathrm{TOL} \\
&\leq (L_1 + L_3L_{y'}L_\sigma)\beta(t) + L_3\gamma(t) + \mathrm{TOL},
\end{aligned}$$

for $t_0 \leq t \leq t_F$. Suppose that $\gamma(t) = \|z'(s) - y'(s)\|$ for $t_0 \leq s \leq t$. Then, from the above inequality, we find

$$\begin{aligned}
\gamma(t) = \|z'(s) - y'(s)\| &\leq (L_1 + L_3L_{y'}L_\sigma)\beta(s) + L_3\gamma(s) + \mathrm{TOL} \\
&\leq (L_1 + L_3L_{y'}L_\sigma)\beta(t) + L_3\gamma(t) + \mathrm{TOL}.
\end{aligned}$$

As $0 < L_3 < 1$, the above inequality becomes

$$\gamma(t) \leq \frac{1}{1-L_3}\left[(L_1 + L_3L_{y'}L_\sigma)\beta(t) + \mathrm{TOL}\right]. \qquad (3.14)$$

Letting $L = L_1 + L_3L_{y'}L_\sigma$ and substituting the above inequality into (3.13) leads to

$$\begin{aligned}
\alpha(t) &\leq \int_{t_0}^{t}\left[L\beta(x) + \frac{L_3}{1-L_3}\left(L\beta(x) + \mathrm{TOL}\right)\right]dx + (t_F - t_0)\mathrm{TOL} \\
&\leq \frac{L}{1-L_3}\int_{t_0}^{t}\beta(x)dx + \frac{1}{1-L_3}(t_F - t_0)\mathrm{TOL}.
\end{aligned}$$

Since the right-hand side of the above inequality is monotone increasing, we have

$$\begin{aligned}
\alpha(t) \leq \beta(t) = \max_{t_0 \leq s \leq t}\alpha(s) = \alpha(s^*) &\leq \frac{L}{1-L_3}\int_{t_0}^{s^*}\beta(x)dx + \frac{1}{1-L_3}(t_F - t_0)\mathrm{TOL} \\
&\leq \frac{L}{1-L_3}\int_{t_0}^{t}\beta(x)dx + \frac{1}{1-L_3}(t_F - t_0)\mathrm{TOL},
\end{aligned}$$

where $t_0 \le s^* \le t$.

Applying Lemma 2 to the above inequality with $K_1 = \dfrac{1}{1-L_3}(t_F - t_0)\text{TOL}$ and $K_2 = \dfrac{L}{1-L_3}$, we obtain

$$\beta(t) \le \frac{1}{1-L_3}(t_F - t_0)\text{TOL}\exp\left[\frac{L}{1-L_3}(t_F - t_0)\right]. \tag{3.15}$$

We hence conclude that

$$\begin{aligned} \|z(t) - y(t)\| &\le \frac{1}{1-L_3}(t_F - t_0)\text{TOL}\exp\left[\frac{L}{1-L_3}(t_F - t_0)\right] \\ &= C \cdot \text{TOL}, \end{aligned}$$

where $C = \dfrac{1}{1-L_3}(t_F - t_0)\exp\left[\dfrac{L}{1-L_3}(t_F - t_0)\right]$.

□

Remark 1: Theorem 6 tells us that the global error associated with the method for NDEs is also bounded by a multiple of tolerance on the whole interval as long as the defect is bounded by tolerance even when the order of the defect is less than p. However, whenever the order of the defect is less than p, the stepsize is taken smaller to ensure the defect is bounded by tolerance. This indicates that while the order of the defect reflects the potential efficiency of a step, the occasional step at lower than the optimal order will not affect the reliability — only the efficiency of the method.

Remark 2: It follows from (3.14) and (3.15) that the global error in the approximation to $y'(t)$ is also bounded by a multiple of TOL.

Chapter 4

Numerical Experiments

4.1 A Problem Set

The problems described below are taken from (Paul 1994) except for Problems 4 and 8. The original source for each problem is also indicated. We created Problem 8 by modifying Problem 7. The first four problems are RDEs and the last four are NDEs. We note that all the NDEs (Problems 5, 6, 7, and 8) have delays in both their solutions and derivatives.

Problem 1: Model of chronic granulocytic leukemia (Wheldon, Kirk and Finlay 1974)

$$\begin{aligned} y_1'(t) &= \frac{1.1}{1+\sqrt{10}(y_1(t-20))^{5/4}} - \frac{10y_1(t)}{1+40y_2(t)}, \\ y_2'(t) &= \frac{100y_1(t)}{1+40y_2(t)} - 2.43y_2(t), \end{aligned}$$

for $0 \le t \le 100$. The initial condition is

$$\phi_1(t) = \frac{1.05767027}{3}, \quad \phi_2(t) = \frac{1.030713491}{3}, \quad \text{for} \quad t \le 0.$$

- No analytical solution available
- A system of RDEs with a constant delay
- First order derivative discontinuity at the initial point
- No vanishing delay

Problem 2: Model of antigen antibody dynamics with fading memory (Gatica and Waltman 1982)

$$\begin{aligned}
y_1'(t) &= -r_1 y_1(t) y_2(t) + r_2 y_3(t), \\
y_2'(t) &= -r_1 y_1(t) y_2(t) + \alpha r_1 y_1(t - y_4(t)) y_2(t - y_4(t)), \\
y_3'(t) &= r_1 y_1(t) y_2(t) - r_2 y_3(t), \\
y_4'(t) &= 1 + \frac{3\delta - y_1(t) y_2(t) - y_3(t)}{y_1(t - y_4(t)) y_2(t - y_4(t)) + y_3(t - y_4(t))} \exp(\delta y_4(t)),
\end{aligned}$$

for $0 \le t \le 40$ with $r_1 = 0.02, r_2 = 0.005, \alpha = 3$, and $\delta = 0.01$. The initial condition is

$$\phi_1(t) = 5, \quad \phi_2(t) = 0.1, \quad \phi_3(t) = \phi_4(t) = 0, \quad \text{for} \quad t \le 0.$$

- No analytical solution available
- A system of RDEs with a state dependent delay
- First order derivative discontinuity at the initial point
- A vanishing delay at $t = 0$

Problem 3: Artificial problem (Neves 1975b)

$$\begin{aligned}
y'(t) &= 1 - y\left(\exp\left(1 - \frac{1}{t}\right)\right), \quad \text{for} \quad 0.1 \le t \le 10, \\
\phi(t) &= \ln(t), \quad \text{for} \quad 0 < t \le 0.1.
\end{aligned}$$

- Analytical solution: $y(t) = \ln(t)$
- A scalar RDE with a time dependent delay
- No discontinuity
- A vanishing delay at $t = 1$

Problem 4: Artificial problem (Neves 1975b)

$$\begin{aligned}
y_1'(t) &= y_2(t), \\
y_2'(t) &= -y_2(\exp(1 - y_2(t))) y_2^2(t) \exp(1 - y_2(t)),
\end{aligned}$$

for $0.1 \leq t \leq 5$. The initial condition is

$$\phi_1(t) = \ln(t), \quad \phi_2(t) = \frac{1}{t}, \quad \text{for} \quad 0 < t \leq 0.1.$$

- Analytical solution: $y_1(t) = \ln(t), y_2(t) = 1/t$
- A system of RDEs with a state dependent delay
- No discontinuity
- A vanishing delay at $t = 1$

Problem 5: Model of food-limited population (Kuang and Feldstein 1991)

$$\begin{aligned} y'(t) &= ry(t)(1 - y(t-1) - cy'(t-1)), \quad \text{for} \quad 0 \leq t \leq 40, \\ \phi(t) &= 2 + t, \quad \text{for} \quad t \leq 0, \end{aligned}$$

where $r = \pi/\sqrt{3} + 1/20$ and $c = \sqrt{3}/2\pi - 1/25$.

- No analytical solution available
- A scalar NDE with a constant delay
- First order derivative discontinuity at the initial point since

$$\left(\lim_{h \to 0+} (f(h, y(h), y(h - \sigma))) = -0.878 \cdots \right) \neq (\phi'(0) = 1)$$

- No vanishing delay

Problem 6: Logistic Gauss-type predator-prey systems (Kuang 1991)

$$\begin{aligned} y_1'(t) &= y_1(t)(1 - y_1(t-\tau) - \rho y_1'(t-\tau)) - \frac{y_2(t)y_1(t)^2}{y_1(t)^2 + 1}, \\ y_2'(t) &= y_2(t)\left(\frac{y_1(t)^2}{y_1(t)^2 + 1} - \alpha\right), \end{aligned}$$

for $0 \leq t \leq 2$ with $\alpha = 1/10, \rho = 29/10$, and $\tau = 21/50$. The initial condition is

$$\phi_1(t) = \frac{33}{100} - \frac{1}{10}t, \quad \phi_2(t) = \frac{111}{50} + \frac{1}{10}t, \quad \text{for} \quad t \leq 0.$$

- No analytical solution available

- A system of NDEs with a constant delay
- First order derivative discontinuity at the initial point since

$$\begin{aligned}(\lim_{h\to 0^+}(f_1(h, y(h), y(h-\sigma))) = 0.0849\cdots) &\neq (\phi_1'(0) = -0.1)\,,\\ (\lim_{h\to 0^+}(f_2(h, y(h), y(h-\sigma))) = -0.0398\cdots) &\neq (\phi_2'(0) = 0.1)\end{aligned}$$

- No vanishing delay

Problem 7: Artificial problem (Castleton and Grimm 1973)

$$y'(t) = \cos(t)(1 + y(ty^2(t))) + y(t)y'(ty^2(t)) - \sin(t + t\sin^2(t)),$$

for $0 \le t \le 1.4$. The initial condition is

$$\phi(0) = 0, \quad \phi'(0) = 1.$$

- Analytical solution: $y(t) = \sin(t)$
- A scalar NDE with a state dependent delay
- No discontinuity since $ty^2(t) = t\sin^2(t) \ge 0$ for any $t \ge 0$
- A vanishing delay at $t = 0$
- Singularity (and a vanishing delay) at $t = \pi/2 = 1.57\cdots$

Problem 8: Modified version of Problem 7

$$\begin{aligned} y'(t) &= \cos(t)(1 + y(ty^2(t))) + L_3 y(t)y'(ty^2(t)) \\ &\quad + (1 - L_3)\sin(t)\cos(t\sin^2(t)) - \sin(t + t\sin^2(t)), \end{aligned}$$

for $0 \le t \le \pi$ with $L_3 = 0.1, 0.3, 0.5^\dagger$. The initial condition is

$$\phi(0) = 0, \quad \phi'(0) = 1.$$

- Analytical solution: $y(t) = \sin(t)$
- A scalar NDE with a state dependent delay

$^\dagger L_3$ is a Lipschitz constant with respect to y' at $\pi/2$.

- No discontinuity
- Vanishing delays at $t = 0, t = \pi/2$, and $t = \pi$

Note that Problem 8 is an NDE and the delay vanishes at $t = \pi/2$ that is in the interior of integration interval.

4.2 A Description of Codes

The experimental code we have developed was implemented as a modified version of DVERK (Hull, Enright and Jackson 1976) and based on the formula CVSS6B (Enright 1993), which is a 6th order CRK formula. We call the code DDVERK. See Appendix A for the prologue of DDVERK and Appendix B for a sample program to solve Problem 7.

The estimate of the defect we used is sampled at $\tau = 0.64$ (which was justified in (Enright 1993) for SDEs) and is the maximum norm of the weighted defect. The weight for the jth component is $1/(\max(1, |y_j'(t)|))$.

The minimum stepsize, Hmin, is taken as 10^{-6} unless otherwise specified. As discussed in §3.2.2, there can be order reductions on some steps if we use the special interpolant for NDEs. Accordingly, a smaller stepsize will likely be selected for those steps. If we are restarting after encountering a delay equal to zero, then order reduction is likely and we take the default initial stepsize to be $100 \cdot \text{Hmin}$. Then the factor the stepsize is allowed to increase by is changed to 50 from 2 until a rejected step occurs.

To construct interpolants to compute delay values in past steps, we save the solutions and the derivatives at the endpoints of each step together with some internal derivative values. The size of the storage space for the information depends on the particular problem. Hence, the user must specify the size of the storage space. If the code requires information beyond the available stored information, it simply returns an error message.

The stopping criterion for the iteration procedure (Algorithm 3) is

$$\max_i |k_i^{(l)} - k_i^{(l-1)}| \leq \text{TOL},$$

and the maximum number of iterations is set to be 5. This implies that we perform at least one iteration.

For a discussion of the discontinuity handling see §2.2.

For comparison, we solved the same problem set using available codes whose descriptions are as follows.

ARCHI (Paul 1992b)

- **The underlying formula** : based on the 5th order Dormand and Prince explicit RK formula with a quartic 5th order interpolant.
- **Discontinuity handling**: there are two options, either tracking or no tracking. To use the tracking option, the user must specify the dependency of derivative discontinuities among components of a DDE, the location of any derivative discontinuities in the initial interval, and the order of the derivative discontinuities. The code then tries to locate propagated discontinuities by finding zeros of the associated nonlinear equations based on the algorithm proposed by Willé and Baker (Willé and Baker 1992). With the 'no tracking' option, the error control mechanism of the underlying formula is relied on to handle discontinuities. We report results with the tracking option.
- **Small delays and vanishing delays**: there are two options, either extrapolation or refinement. The refinement option carries out the procedure described in §1.3.1.
- **The error estimate**: we chose the infinite-norm continuous error estimator among five available options for an error control mechanism. With this option, the code computes the difference between a 5th order interpolant and a 4th order interpolant as a continuous asymptotic error polynomial. Then it estimates the maximum continuous local error by manipulating this polynomial.
- **The starting stepsize**: the user must supply the starting stepsize. We used 0.1 for the starting stepsize for all problems but Problem 4. The starting stepsize for Problem 4 is described in the next section.

DRKLAG (Neves and Thompson 1992b)

- **The underlying formula**: based on a continuously embedded RK formula pair 5(4) of Sarafyan with a cubic 4th order interpolant.
- **Discontinuity handling**: there are two options, either rootfinding or ignoring. With the rootfinding option, the code assumes that a derivative discontinuity occurs at the initial point $t = t_0$. The code then tries to locate propagated discontinuities by finding zeros of the associated nonlinear equations.

- **Small delays and vanishing delays**: Tavernini's method (Tavernini 1971) is used in the first step. In other steps, there are two options, either restricting the stepsize or using extrapolation. With the extrapolation option, the code still restricts the amount the stepsize may increase on each step. As a consequence, the number of steps taken can be large. We chose the first option for non-vanishing delay problems (1, 5, 6) and the second option for vanishing delay problems (2, 3, 4, 7).

- **The error estimate**: the code computes the difference between a 5th order RK formula and a 4th order RK formula at the mesh point and then estimates the local error. We used absolute error control.

- **The starting stepsize**: we took the default starting stepsize except for Problem 4.

- The increment of the independent variable (which must be supplied by the user) to identify output points was taken as the length of interval.

4.3 Numerical Results

In this section we present numerical results by summarizing them in tables and graphs and make some observations about the results. All computations were done on SUN Sparcserver 20/612 under the Solaris 2.3 operating system.

4.3.1 A Description of Computations

To compare methods that deliver 'comparable' error, we must take care in selecting the error criterion. Absolute error is not reliable when the magnitude of a solution changes significantly over the range of integration while relative error is not appropriate when a solution is 0 or close to 0. We have chosen to use the absolute error on an appropriately scaled version of each problem. The *scaled* problem is the original problem scaled by a diagonal matrix $D = \mathrm{diag}(1/d_1, 1/d_2, \ldots, 1/d_n)$, where $d_i = \max |y_i(t)|$, so that the maximum magnitude of the solution of the scaled problem is 1. Specifically, by letting $w(t) = Dy(t)$, the scaled problem is

$$\begin{aligned} w'(t) &= Df(t, D^{-1}w(t), D^{-1}w(t - \sigma_1(t, D^{-1}w(t))), D^{-1}w'(t - \sigma_2(t, D^{-1}w(t)))), \\ w(t) &= D\phi(t), \quad w'(t) = D\phi'(t), \end{aligned}$$

for the original (unscaled) problem

$$\begin{aligned} y'(t) &= f(t, y(t), y(t-\sigma_1(t, y(t))), y'(t-\sigma_2(t, y(t)))), \\ y(t) &= \phi(t), \quad y'(t) = \phi'(t). \end{aligned}$$

For Problems 3, 4, 7, and 8 which have analytical solutions we used the maximum magnitude of the analytical solutions over the range of integration for constructing the scaling matrix D. For Problems 1, 2, 5, and 6 we carried out the following three steps to construct the scaling matrix D and to prepare for computing the reference solutions.

Step 1: Solving the original (unscaled) problem by DDVERK with quadruple precision and with tolerance 10^{-12} while saving necessary data to construct an interpolant over the range of integration.

Step 2: Evaluating the interpolant at 100 sample points in each step as an approximation to the analytical solution and constructing the scaling matrix D from the maximal values observed.

Step 3: Solving the scaled problem by DDVERK with quadruple precision and with tolerance 10^{-12} while saving necessary data to construct an interpolant over the range of integration for use as the reference 'true solution'.

The scaled problems were solved by ARCHI, DDVERK, and DRKLAG in double precision with tolerances 10^{-k}, $k = 2, 4, 6, 8, 10$. All the numerical results presented are for these scaled problems.

4.3.2 Numerical Results

The statistics we report in tables and graphs are as follows.

TIME: CPU time in seconds

FCN: total number of derivative evaluations required to solve the scaled problem

STEP: total number of successful steps required to solve the scaled problem

GE1: maximum norm of the absolute global error at the endpoint t_F expressed in units of tolerance TOL

GE2: maximum of the maximum norm of the absolute global error at mesh points expressed in units of tolerance TOL

GE3: maximum of the maximum norm of the absolute global error over the range of integration, computed using 100 sample points in each step, expressed in units of tolerance TOL

DEF1: maximum of the maximum norm of the defect over the range of integration, computed using 100 sample points in each step. (except for the intervals in which the code locates discontinuities) expressed in units of tolerance TOL

DEF2: maximum of the maximum norm of the defect over intervals excluded when computing DEF1 ignoring a very small neighborhood near the identified discontinuities, computed using 100 sample points in each such interval, expressed in units of tolerance TOL

The number $k = 2, 4, 6, 8, 10$ below the tolerance TOL in tables indicates that the tolerance used is 10^{-k}. The numbers below PROB in tables indicates problem number.

Missing entries in Tables indicate that problems were not able to be solved with the corresponding method and tolerance. Note that Problem 8 could not be solved by the version of DRKLAG we obtained.

With ARCHI there are two options for handling small/vanishing delays: extrapolation and refinement. The methods used with these two options are completely different since the code with the refinement option reduces to a collocation method as described in §1.3.1. We report numerical results separately for each option in two tables although the results for Problems 1, 3, 5, and 6 (whose delays never fell in the current step) are exactly the same. On these problems only one result is plotted in the graphs. We use ARCHI_E and ARCHI_R to refer to ARCHI with the extrapolation option and ARCHI with the refinement option respectively.

We do not present the summary statistics for Problems 3, 4, and 7 in Table 4.7 (DRKLAG with rootfinding option), since these problems do not possess discontinuities and the results are the same as in Table 4.6 (DRKLAG without rootfinding option).

The initial stepsizes used for Problem 4 (which could not be solved with default initial

Method	TOL	HINIT
ARCHI_E	10^{-8}	10^{-5}
	10^{-10}	10^{-2}
ARCHI_R	all tolerances	10^{-2}
DDVERK	10^{-4}	10^{-4}
	$10^{-2}, 10^{-6}, 10^{-10}$	10^{-2}
DRKLAG	10^{-6}	10^{-6}
	10^{-8}	10^{-2}

Table 4.1: Initial Stepsizes for Problem 4

TOL	Located Discontinuity Points			
2	0.4235378474	0.8461447914	1.273045603	1.693507362
4	0.4200516464	0.8401757062	1.260178283	1.680190486
6	0.4200004578	0.8400009155	1.260001373	1.680001412
8	0.4200000115	0.8400000168	1.260000018	1.680000037
10	0.4200000000	0.8400000001	1.260000000	1.680000000

Table 4.2: Points where Discontinuities were Located by DDVERK for Problem 6

stepsizes as specified in §4.1) are presented in Table 4.1.

To illustrate how the automatic handling technique for discontinuities works for DDEs, we present discontinuity points detected by DDVERK for Problem 6 in Table 4.2. The exact discontinuity points for Problem 6 are $0.42, 0.84, 1.26$, and 1.68.

Defects computed by DDVERK are presented in Table 4.11. Entries which have a value 0 for DEF2 indicate that the code did not locate a discontinuity with the corresponding problem and tolerance.

Since the slope of a $\log_{10}(\textit{the number of steps})$ versus $-\log_{10}(\textit{error})$ graph represents the order of convergence, we approximated the slope in a least square sense over prescribed tolerances. They are presented in Tables 4.12 and 4.13. In there, we used GE3∗TOL to compute the observed orders of convergence for global error and DEF1∗TOL to compute the observed orders of convergence for defect.

Numerical results are also plotted in Figures 4.1 to 4.8 for comparison purposes. In these figures, the global error is equivalent to GE3∗TOL. The graphs of Problems 1, 2, 5, and 6 for

DRKLAG are with the rootfinding option.

We present plots of Global Error vs. FCN since a conventional way to measure cost is to count the number of derivative evaluations made to solve a problem. However, this measure does not take account of the complexity used to handle discontinuities for ARCHI and DRKLAG with tracking and rootfinding options respectively. These codes try to locate propagated derivative discontinuities by finding zeros of an associated nonlinear equation, and this does not involve derivative evaluations. DDVERK handles discontinuities by a defect based strategy which involves derivative evaluations and the number of derivative evaluations for this task is included in the total number of derivative evaluations.

The plots of TOL vs. Global Error for each problem are intended only to provide a crude measure of the 'tolerance proportionality' that was achieved by the methods on each problem. The closer the 'slope' of these curves are to one the better the performance (the actual values of these curves are otherwise not important in assessing tolerance proportionality).

PROB	TOL	STAT					
		TIME	FCN	STEP	GE1	GE2	GE3
1	2	2.826	1607	110	$2.1029e-02$	$7.7103e-02$	$1.6321e-01$
	4	2.907	2108	184	$3.4682e-02$	$2.6697e-01$	$2.6697e-01$
	6	3.206	3946	349	$3.3915e-02$	$1.1322e-01$	$1.1322e-01$
	8	3.685	7770	693	$4.0061e-02$	$3.6259e+00$	$3.6260e+00$
	10	4.936	15828	1424	$1.8322e-02$	$2.9846e-01$	$2.9853e-01$
2	2	0.713	668	30	$6.1536e-04$	$6.1536e-04$	$6.1536e-04$
	4	0.718	688	31	$6.1221e-02$	$6.1221e-02$	$6.1221e-02$
	6	0.774	969	33	$2.6807e+00$	$2.6807e+00$	$2.6807e+00$
	8	1.120	1737	59	$5.0413e+00$	$5.0413e+00$	$5.0413e+00$
	10	1.446	2995	120	$4.6367e+00$	$4.7687e+00$	$4.8134e+00$
3	2	0.103	252	10	$9.8514e-02$	$1.2328e-01$	$1.2328e-01$
	4	0.102	243	12	$1.1567e-01$	$1.1567e-01$	$1.1567e-01$
	6	0.134	321	18	$2.9529e-01$	$2.9529e-01$	$2.9529e-01$
	8	0.136	508	32	$1.9415e-01$	$1.9415e-01$	$1.9415e-01$
	10	0.186	887	62	$4.7351e-03$	$3.0147e-01$	$3.0155e-01$
4	2	0.127	277	18	$5.1276e-02$	$5.1276e-02$	$5.1276e-02$
	4	0.126	240	15	$5.3069e-02$	$5.4729e-02$	$5.5755e-02$
	6	0.139	381	26	$1.2853e-01$	$1.3957e-01$	$1.4222e-01$
	8	0.175	738	49	$1.1310e-01$	$6.0793e-01$	$6.0793e-01$
	10	0.286	1325	100	$7.2985e-02$	$1.3532e+00$	$1.3533e+00$
5	2	0.844	870	35	$5.6493e+00$	$7.1371e+00$	$9.9070e+00$
	4	0.909	1696	83	$6.5743e+01$	$9.4366e+01$	$1.0102e+02$
	6	1.021	2548	145	$1.5785e+00$	$4.9803e+00$	$4.9803e+00$
	8	1.438	5420	306	$1.1175e+01$	$1.7187e+01$	$1.7187e+01$
	10	2.000	9882	620	$4.7856e+00$	$8.0470e+00$	$8.0470e+00$
6	2	0.179	259	9	$1.8687e-01$	$5.6108e-01$	$5.6108e-01$
	4	0.183	306	10	$5.6972e-01$	$6.9901e-01$	$6.9901e-01$
	6	0.183	231	10	$2.6106e-01$	$5.0721e-01$	$5.0721e-01$
	8	0.199	409	14	$1.1526e+00$	$1.4446e+00$	$1.4446e+00$
	10	0.208	439	19	$2.3123e-01$	$4.2756e-01$	$4.2756e-01$
7	2	0.093	52	3	$6.0449e-03$	$6.0449e-03$	$5.9443e-02$
	4	0.101	143	5	$1.8626e-02$	$2.7644e-02$	$3.6954e-02$
	6	0.105	168	7	$2.4572e-01$	$2.8477e-01$	$3.1474e-01$
	8	0.116	262	14	$6.1105e-01$	$6.6474e-01$	$6.6809e-01$
	10	0.143	506	28	$1.6804e-01$	$1.9072e-01$	$2.3770e-01$

Table 4.3: Summary Statistics for DDVERK (Problems 1 to 7)

PROB	TOL	STAT					
		TIME	FCN	STEP	GE1	GE2	GE3
1	2	2.872	1309	157	$4.2957e-04$	$2.3145e-01$	$2.5862e-01$
	4	2.871	1925	238	$4.8018e-03$	$3.4202e-01$	$3.7038e-01$
	6	3.161	4245	524	$5.7117e-03$	$2.5611e-01$	$2.6429e-01$
	8	3.803	10005	1245	$9.0428e-03$	$2.5294e-01$	$2.5657e-01$
	10	5.677	24509	3056	$8.7624e-03$	$1.0336e-01$	$1.0382e-01$
2	2	—	—	—	—	—	—
	4	0.695	561	65	$1.1669e+02$	$1.1669e+02$	$1.1669e+02$
	6	0.997	1385	157	$1.1840e+01$	$5.4900e+01$	$5.5027e+01$
	8	2.763	4521	494	$6.1819e+01$	$8.0567e+01$	$8.0818e+01$
	10	13.106	13145	1587	$1.5881e+02$	$2.3302e+02$	$2.3364e+02$
3	2	0.100	169	14	$9.9068e+00$	$9.9068e+00$	$9.9068e+00$
	4	0.108	137	16	$1.0390e-01$	$1.0189e+00$	$1.2844e+00$
	6	0.108	273	32	$3.1231e-03$	$3.1808e-02$	$5.4820e-02$
	8	0.129	617	75	$3.3376e-03$	$2.8585e-02$	$3.0764e-02$
	10	0.184	1489	183	$1.5283e-02$	$2.5306e-02$	$2.6240e-02$
4	2	—	—	—	—	—	—
	4	—	—	—	—	—	—
	6	—	—	—	—	—	—
	8	0.262	1937	231	$5.6746e+01$	$5.6746e+01$	$5.6746e+01$
	10	0.628	6073	752	$1.6462e+02$	$1.6462e+02$	$1.6462e+02$
5	2	0.896	576	64	$6.3369e-01$	$6.4429e-01$	$7.7719e-01$
	4	1.068	1136	130	$1.5002e-01$	$2.1355e-01$	$2.2091e-01$
	6	1.391	2368	276	$2.1539e-01$	$2.9827e-01$	$3.0900e-01$
	8	2.165	5368	648	$1.1199e-01$	$2.2205e-01$	$2.2234e-01$
	10	4.446	12872	1590	$3.4146e-01$	$4.9085e-01$	$4.9186e-01$
6	2	0.142	53	6	$3.1343e-06$	$4.6483e-06$	$9.5991e-06$
	4	0.150	53	6	$3.1342e-04$	$4.6483e-04$	$9.5991e-04$
	6	0.154	85	9	$2.3047e-02$	$3.8014e-02$	$4.3144e-02$
	8	0.156	149	16	$4.5712e-03$	$1.2373e-02$	$2.8561e-02$
	10	0.177	293	34	$7.2564e-03$	$4.3555e-02$	$4.8400e-02$
7	2	0.093	57	3	$1.6149e+00$	$1.6149e+00$	$1.6695e+00$
	4	0.093	105	7	$2.4985e+00$	$2.4985e+00$	$2.4985e+00$
	6	0.105	193	17	$4.7672e+00$	$7.3692e+00$	$7.3692e+00$
	8	0.130	497	49	$3.4502e+01$	$3.6831e+01$	$3.7059e+01$
	10	0.216	1545	161	$1.0495e+02$	$1.1027e+02$	$1.1036e+02$

Table 4.4: Summary Statistics for ARCHI with extrapolation option (Problems 1 to 7)

PROB	TOL	STAT TIME	FCN	STEP	GE1	GE2	GE3
1	2	2.804	1309	157	$4.2957e-04$	$2.3145e-01$	$2.5862e-01$
	4	2.875	1925	238	$4.8018e-03$	$3.4202e-01$	$3.7038e-01$
	6	3.078	4245	524	$5.7117e-03$	$2.5611e-01$	$2.6429e-01$
	8	3.851	10005	1245	$9.0428e-03$	$2.5294e-01$	$2.5657e-01$
	10	5.269	24509	3056	$8.7624e-03$	$1.0336e-01$	$1.0382e-01$
2	2	0.607	281	7	$2.9500e-01$	$2.9500e-01$	$2.9500e-01$
	4	0.638	481	12	$4.9940e-03$	$1.4355e-02$	$4.8599e-02$
	6	0.749	1041	26	$2.7964e-02$	$2.8537e-02$	$3.8137e-02$
	8	1.034	2410	60	$2.2451e-02$	$2.3188e-02$	$2.3309e-02$
	10	1.744	6068	147	$5.8598e-03$	$4.0294e-02$	$4.0321e-02$
3	2	0.097	197	8	$6.1425e-03$	$6.1425e-03$	$6.1425e-03$
	4	0.105	293	15	$5.7397e-02$	$8.5965e-02$	$1.5880e-01$
	6	0.112	572	32	$3.0560e-02$	$3.1808e-02$	$5.4820e-02$
	8	0.148	1117	75	$2.4455e-02$	$2.8585e-02$	$3.0764e-02$
	10	0.212	2283	183	$1.8844e-02$	$2.5306e-02$	$2.6240e-02$
4	2	—	—	—	—	—	—
	4	0.146	512	37	$7.5951e-01$	$7.5951e-01$	$1.4590e+00$
	6	0.172	896	73	$1.2310e+01$	$1.2310e+01$	$1.2310e+01$
	8	0.288	2208	206	$6.1158e+01$	$6.1158e+01$	$6.1158e+01$
	10	0.653	6483	723	$1.5659e+02$	$1.5659e+02$	$1.5659e+02$
5	2	0.947	576	64	$6.3369e-01$	$6.4429e-01$	$7.7719e-01$
	4	1.089	1136	130	$1.5002e-01$	$2.1355e-01$	$2.2091e-01$
	6	1.390	2368	276	$2.1539e-01$	$2.9827e-01$	$3.0900e-01$
	8	2.186	5368	648	$1.1199e-01$	$2.2205e-01$	$2.2234e-01$
	10	4.161	12872	1590	$3.4146e-01$	$4.9085e-01$	$4.9186e-01$
6	2	0.105	53	6	$3.1343e-06$	$4.6483e-06$	$9.5991e-06$
	4	0.101	53	6	$3.1342e-04$	$4.6483e-04$	$9.5991e-04$
	6	0.109	85	9	$2.3047e-02$	$3.8014e-02$	$4.3144e-02$
	8	0.138	149	16	$4.5712e-03$	$1.2373e-02$	$2.8561e-02$
	10	0.253	293	34	$7.2564e-03$	$4.3555e-02$	$4.8400e-02$
7	2	0.088	125	3	$5.3428e-04$	$5.3428e-04$	$5.3428e-04$
	4	0.097	170	4	$1.2923e-02$	$1.2923e-02$	$1.2923e-02$
	6	0.108	193	17	$4.7672e+00$	$7.3692e+00$	$7.3692e+00$
	8	0.139	497	49	$3.4502e+01$	$3.6831e+01$	$3.7059e+01$
	10	0.223	1545	161	$1.0495e+02$	$1.1027e+02$	$1.1036e+02$

Table 4.5: Summary Statistics for ARCHI with refinement option (Problems 1 to 7)

PROB	TOL	STAT					
		TIME	FCN	STEP	GE1	GE2	GE3
1	2	2.178	902	148	$7.9609e-02$	$2.4960e-01$	$2.1390e+00$
	4	2.248	1232	202	$3.0925e-01$	$2.8922e+00$	$4.0308e+00$
	6	2.524	2672	437	$2.9176e-01$	$1.7781e+00$	$2.9030e+00$
	8	3.311	6260	1032	$6.8795e-01$	$5.1310e+00$	$5.1341e+00$
	10	5.532	15260	2520	$2.5651e-01$	$1.5142e+00$	$3.2496e+00$
2	2	0.646	705	97	$5.3778e-01$	$5.3778e-01$	$5.3778e-01$
	4	0.844	1261	157	$4.8201e-01$	$1.5429e+00$	$1.5436e+00$
	6	1.399	2953	355	$4.8621e-01$	$8.7586e+00$	$8.7593e+00$
	8	2.669	5992	870	$1.2559e+00$	$3.2907e+01$	$3.2908e+01$
	10	5.872	13180	2168	$6.1241e+00$	$1.6152e+02$	$1.6152e+02$
3	2	0.099	128	16	$1.8049e-02$	$1.8049e-02$	$1.8049e-02$
	4	0.109	244	34	$8.0700e-04$	$8.0700e-04$	$9.2239e-04$
	6	0.134	511	78	$1.2928e-03$	$1.2928e-03$	$1.2928e-03$
	8	0.203	1188	189	$6.1937e-04$	$6.1937e-04$	$6.2030e-04$
	10	0.429	2866	466	$1.6653e-04$	$1.6653e-04$	$1.6875e-04$
4	2	0.116	202	25	$3.4194e-02$	$3.4883e-02$	$3.5135e-02$
	4	—	—	—	—	—	—
	6	0.167	580	92	$3.8549e-02$	$3.9240e-02$	$4.0202e-02$
	8	0.282	1185	190	$4.5960e-02$	$4.6770e-02$	$4.8232e-02$
	10	0.577	2679	436	$4.4884e-02$	$4.5676e-02$	$4.7709e-02$
5	2	0.647	505	57	$1.8755e+00$	$4.7156e+00$	$5.8562e+00$
	4	0.689	777	90	$1.0439e+01$	$1.6916e+01$	$2.4987e+01$
	6	0.906	2101	241	$5.6659e+00$	$2.0865e+01$	$3.3282e+01$
	8	1.321	4963	608	$6.7105e+01$	$1.0917e+02$	$1.1040e+02$
	10	2.169	11119	1467	$1.5644e+01$	$3.2153e+01$	$3.7404e+01$
6	2	0.146	78	7	$7.1352e-01$	$1.4120e+00$	$2.4466e+00$
	4	0.174	284	20	$6.8217e-01$	$1.7254e+01$	$2.8989e+01$
	6	0.280	836	59	$1.0308e+01$	$1.0641e+01$	$1.4516e+01$
	8	0.346	1202	96	$1.5502e+01$	$1.9582e+01$	$2.1556e+01$
	10	0.401	1694	152	$1.4998e+01$	$1.4998e+01$	$1.8038e+01$
7	2	0.100	152	10	$1.7393e-01$	$2.3254e-01$	$2.3388e-01$
	4	0.108	227	19	$5.4070e-01$	$7.0826e-01$	$7.1587e-01$
	6	0.126	402	41	$1.3725e+00$	$1.7823e+00$	$1.7880e+00$
	8	0.176	757	92	$1.5676e+00$	$2.0540e+00$	$2.0563e+00$
	10	0.281	1507	211	$2.5905e+00$	$3.3988e+00$	$3.3990e+00$

Table 4.6: Summary Statistics for DRKLAG without rootfinding option (Problems 1 to 7)

PROB	TOL	STAT					
		TIME	FCN	STEP	GE1	GE2	GE3
1	2	2.259	954	151	$7.7785e-02$	$3.4172e-01$	$2.9557e+00$
	4	2.311	1272	206	$4.6250e-02$	$3.6848e-01$	$3.1761e+00$
	6	2.657	2682	439	$6.6659e-02$	$3.0012e-01$	$3.1603e+00$
	8	3.566	6222	1030	$8.5977e-02$	$1.6029e-01$	$2.9263e+00$
	10	5.855	15156	2516	$8.2447e-02$	$1.1704e-01$	$3.2496e+00$
2	2	0.659	705	97	$5.3778e-01$	$5.3778e-01$	$5.3778e-01$
	4	0.865	1261	157	$4.8201e-01$	$1.5429e+00$	$1.5436e+00$
	6	1.482	2953	355	$4.8621e-01$	$8.7586e+00$	$8.7593e+00$
	8	2.938	5992	870	$1.2559e+00$	$3.2907e+01$	$3.2908e+01$
	10	6.482	13180	2168	$6.1241e+00$	$1.6152e+02$	$1.6152e+02$
5	2	0.741	532	80	$3.2867e-01$	$9.6810e-01$	$1.7857e+00$
	4	0.815	964	138	$2.5340e+00$	$2.3831e+01$	$2.3831e+01$
	6	0.961	2056	269	$3.0053e+01$	$1.0148e+02$	$1.0148e+02$
	8	1.303	4630	597	$1.4757e+01$	$2.6681e+01$	$2.6681e+01$
	10	2.245	10084	1405	$4.2561e+01$	$6.8016e+01$	$6.9984e+01$
6	2	0.165	101	11	$1.3928e-01$	$4.9840e-01$	$9.5429e-01$
	4	0.200	255	21	$8.0015e+00$	$1.0729e+01$	$1.5633e+01$
	6	0.225	474	33	$1.9389e+00$	$1.2210e+01$	$1.3679e+01$
	8	0.272	636	51	$1.9668e+00$	$1.2138e+01$	$1.3604e+01$
	10	0.342	822	77	$1.1865e+01$	$1.3479e+01$	$1.3723e+01$

Table 4.7: Summary Statistics for DRKLAG with rootfinding option (Problems 1, 2, 5, 6)

L_3	TOL	STAT					
		TIME	FCN	STEP	GE1	GE2	GE3
0.1	2	0.104	97	6	$7.4798e-03$	$5.6200e-02$	$2.5403e-01$
	4	0.110	136	6	$2.2028e-01$	$2.2028e-01$	$2.2578e-01$
	6	0.118	282	12	$2.6793e-03$	$1.7553e-01$	$2.3379e-01$
	8	0.138	362	20	$5.1351e-03$	$1.6700e-01$	$1.7890e-01$
	10	0.174	633	40	$2.3896e-02$	$2.0497e-01$	$2.0953e-01$
0.3	2	0.126	118	6	$2.5523e-06$	$4.4408e-02$	$1.7161e-01$
	4	0.128	164	6	$1.2130e-01$	$8.1206e-01$	$8.1206e-01$
	6	0.137	339	12	$1.7583e-02$	$2.0935e-01$	$2.9363e-01$
	8	0.189	698	28	$8.8351e-01$	$1.1305e+01$	$1.1305e+01$
	10	0.205	898	42	$1.2141e-02$	$1.0389e+00$	$1.1076e+00$
0.5	2	0.109	118	6	$8.7111e-03$	$1.3479e-02$	$2.1082e-01$
	4	0.127	250	8	$2.9758e-02$	$4.4224e-01$	$4.7528e-01$
	6	0.141	373	13	$1.5549e-03$	$4.8313e-01$	$5.0475e-01$
	8	0.171	652	27	$5.7775e-01$	$2.7679e+00$	$2.8659e+00$
	10	0.196	849	49	$1.2199e-01$	$8.7889e-01$	$8.8220e-01$

Table 4.8: Summary Statistics for DDVERK (Problem 8)

L_3	TOL	STAT					
		TIME	FCN	STEP	GE1	GE2	GE3
0.1	2	0.112	129	9	$1.3346e+00$	$2.7753e+00$	$3.0787e+00$
	4	0.112	145	12	$2.5573e+00$	$8.9051e+00$	$9.8142e+00$
	6	0.127	345	32	$1.8795e+01$	$5.5346e+01$	$5.5781e+01$
	8	0.191	921	93	$3.8947e+01$	$1.1398e+02$	$1.1407e+02$
	10	1.511	13521	1606	$1.4829e+03$	$5.8753e+03$	$5.8756e+03$
0.3	2	0.096	113	8	$2.2961e-01$	$1.9098e+00$	$2.0023e+00$
	4	0.114	225	18	$7.4730e+00$	$4.4511e+01$	$4.4522e+01$
	6	—	—	—	—	—	—
	8	0.685	5345	608	$2.5842e+02$	$1.4687e+03$	$1.4690e+03$
	10	2.728	26689	3191	$1.1198e+03$	$6.3479e+03$	$6.3483e+03$
0.5	2	—	—	—	—	—	—
	4	0.114	233	18	$1.0828e+01$	$4.5050e+01$	$4.5186e+01$
	6	0.224	1273	128	$2.9546e+01$	$3.5423e+02$	$3.5439e+02$
	8	0.913	7769	808	$1.1564e+02$	$1.4209e+03$	$1.4213e+03$
	10	5.973	53441	5160	$2.5043e+02$	$3.3265e+03$	$3.3267e+03$

Table 4.9: Summary Statistics for ARCHI with extrapolation option (Problem 8)

L_3	TOL	STAT					
		TIME	FCN	STEP	GE1	GE2	GE3
	2	0.107	156	6	$3.7981e-02$	$3.7981e-02$	$8.7501e-02$
	4	0.121	223	9	$7.6276e-01$	$9.6151e-01$	$1.4717e+00$
0.1	6	0.131	403	28	$3.6931e+00$	$6.5540e+00$	$6.5635e+00$
	8	0.327	941	79	$1.6658e+01$	$2.5935e+01$	$2.5939e+01$
	10	0.610	2874	271	$3.8802e+01$	$7.5431e+01$	$7.5624e+01$
	2	0.114	172	8	$9.0842e-02$	$1.3421e-01$	$4.5185e-01$
	4	0.123	239	10	$5.6509e-01$	$6.3634e-01$	$8.3566e-01$
0.3	6	0.149	471	34	$3.2008e+00$	$8.6377e+00$	$8.6609e+00$
	8	0.199	1143	97	$1.1640e+01$	$2.9795e+01$	$2.9804e+01$
	10	0.598	3204	318	$3.9996e+01$	$9.2911e+01$	$9.3185e+01$
	2	0.115	167	4	$1.4159e-02$	$1.4159e-02$	$1.6361e-02$
	4	0.135	279	14	$2.6800e-01$	$7.6811e-01$	$1.5014e+00$
0.5	6	0.151	576	39	$2.4686e+00$	$6.0040e+00$	$6.0236e+00$
	8	0.239	1363	116	$9.3127e+00$	$2.6035e+01$	$2.6438e+01$
	10	0.614	3696	375	$3.2083e+01$	$8.6679e+01$	$8.6999e+01$

Table 4.10: Summary Statistics for ARCHI with refinement option (Problem 8)

PROB	Defect	Tolerance				
		10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
1	DEF1	$1.01e+00$	$9.46e-01$	$8.42e-01$	$1.44e+00$	$2.06e+00$
	DEF2	$0.00e+00$	$1.49e+00$	$1.05e-02$	$5.55e+01$	$8.55e+00$
2	DEF1	$6.56e-04$	$3.00e-02$	$5.45e-01$	$1.18e+00$	$1.82e+00$
	DEF2	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$
3	DEF1	$2.52e+00$	$3.86e+00$	$1.11e+01$	$2.00e+01$	$1.81e+01$
	DEF2	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$	$0.00e+00$
4	DEF1	$8.27e-02$	$6.68e-01$	$1.92e+00$	$5.69e+00$	$4.84e+00$
	DEF2	$0.00e+00$	$0.00e+00$	$0.00e+00$	$5.76e+01$	$2.21e+02$
5	DEF1	$5.36e+00$	$2.88e+02$	$1.21e+02$	$1.12e+02$	$2.89e+02$
	DEF2	$1.33e+00$	$6.20e+02$	$1.86e+01$	$4.08e+02$	$1.11e+02$
6	DEF1	$1.31e-06$	$7.19e-05$	$4.43e-04$	$3.33e-01$	$3.03e+00$
	DEF2	$1.43e-06$	$2.70e-03$	$2.76e-01$	$3.85e+00$	$8.49e+00$
7	DEF1	$1.61e-01$	$1.92e-01$	$8.03e-01$	$1.91e+01$	$5.77e+01$
	DEF2	$0.00e+00$	$0.00e+00$	$0.00e+00$	$6.73e+00$	$5.54e+00$

Table 4.11: DEF1 and DEF2 for DDVERK (Problems 1 to 7)

STAT	Problem 1	2	3	4	5	6	7
Global Error	6.53	6.87	9.07	6.91	6.76	21.95	7.31
Defect	6.82	7.19	8.42	7.05	5.46	3.35	5.25

Table 4.12: Observed Orders of Convergence for DDVERK (Problems 1 to 7)

STAT	L3 0.1	0.3	0.5
Global Error	8.79	6.79	7.48

Table 4.13: Observed Orders of Convergence for DDVERK (Problem 8)

4.3.3 Discussion

The CPU time required to solve problems is comparable among methods except for Problems 2 and 8 with stringent tolerances. For Problem 2 with TOL= 10^{-10}, ARCHI_E took 9 times and DRKLAG took 4 times more CPU time than DDVERK. For Problem 8 with TOL= 10^{-10} and $L_3 = 0.5$, ARCHI_E took 30 times and ARCHI_R took 3 times more CPU time than DDVERK.

GE2 and GE3 for DDVERK, ARCHI_E, and ARCHI_R are comparable for all problems and for all tolerances. However, GE3 for DRKLAG with the rootfinding option for Problem 1 is a magnitude larger than GE2 for all tolerances. This may be due to a poor choice of an interpolant.

The performance of all methods in terms of the proportionality of global error to tolerance looks fine from the TOL vs. Global Error graphs because the slopes of the graphs are almost one except for some intervals, for example, DDVERK for Problem 2 and ARCHI for Problems 6 and 7.

From the Global Error vs. FCN graphs, we observe that DDVERK needs the least number of derivative evaluations to solve vanishing delay problems (Problems 4, 7, 8). For Problem 3 (that is also a vanishing delay problem) and for discontinuous RDEs (Problems 1, 2), the cost for DDVERK is comparable with the other methods. For discontinuous NDEs (Problems 5, 6),

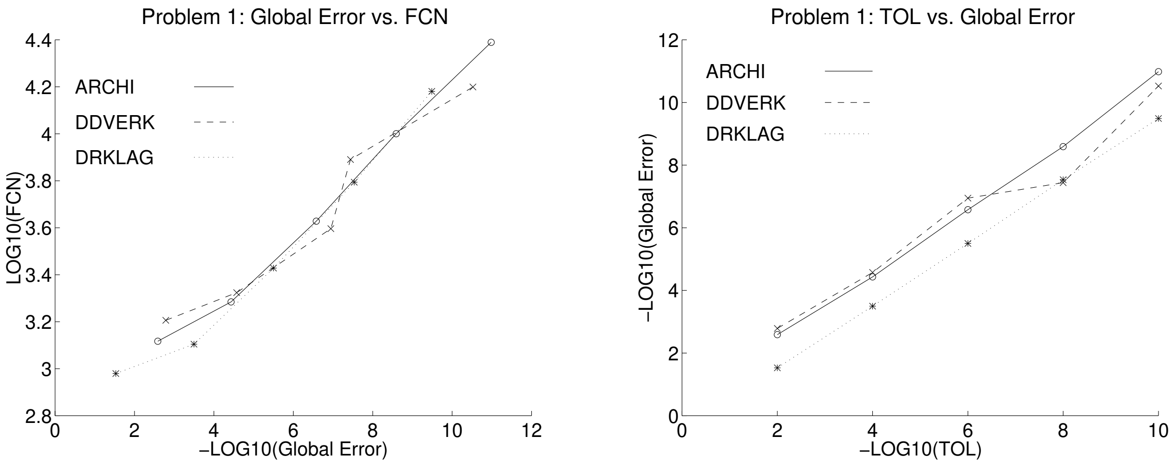

Figure 4.1: Global Error vs. FCN and TOL vs. Global Error Graphs for Problem 1

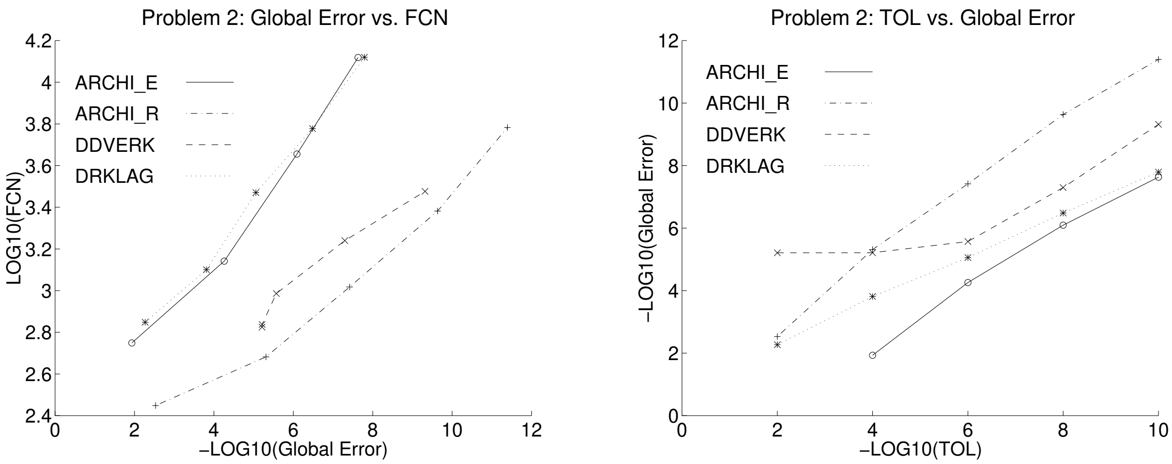

Figure 4.2: Global Error vs. FCN and TOL vs. Global Error Graphs for Problem 2

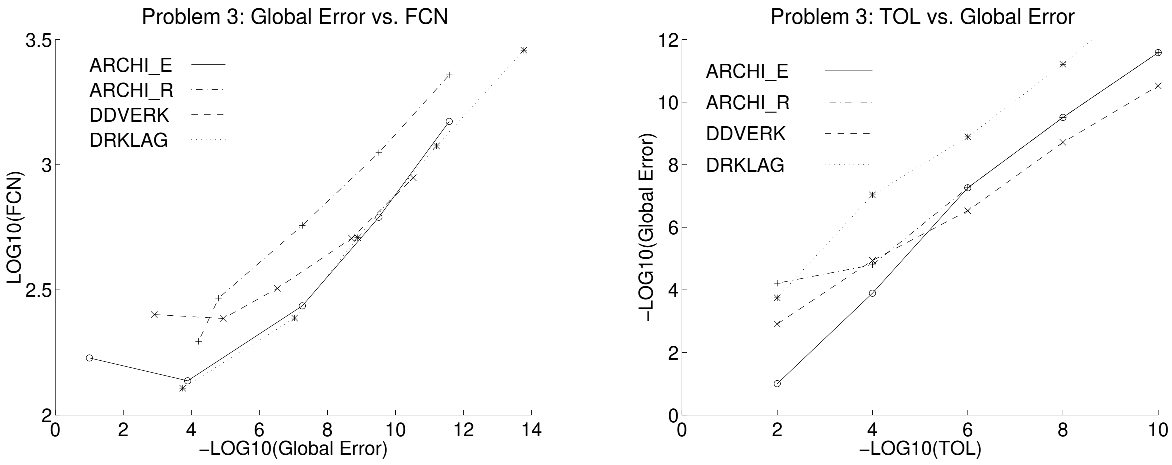

Figure 4.3: Global Error vs. FCN and TOL vs. Global Error Graphs for Problem 3

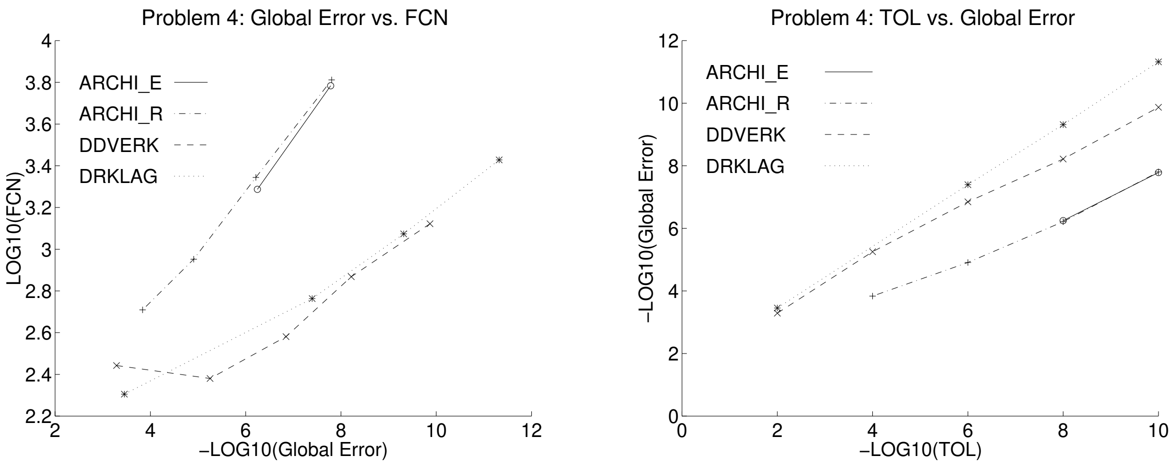

Figure 4.4: Global Error vs. FCN and TOL vs. Global Error Graphs for Problem 4

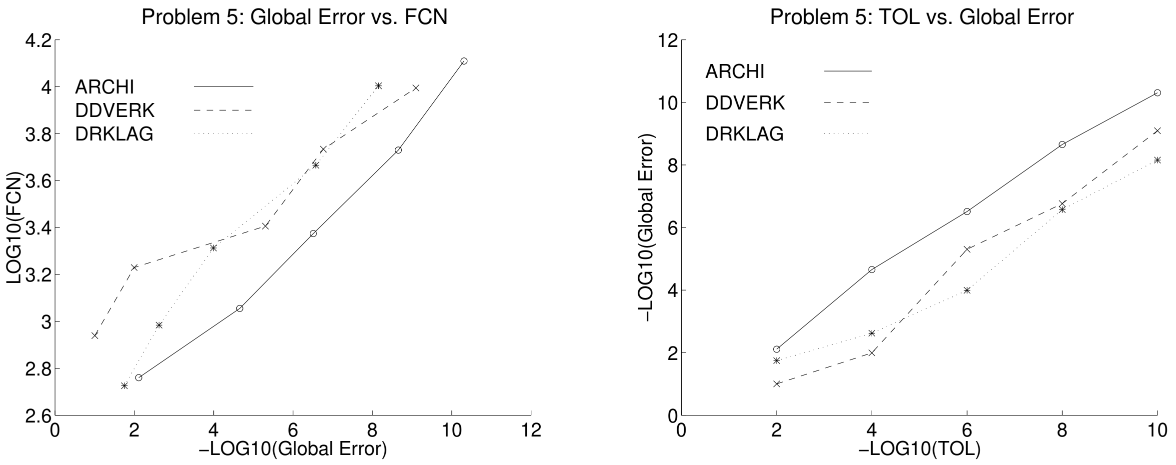

Figure 4.5: Global Error vs. FCN and TOL vs. Global Error Graphs for Problem 5

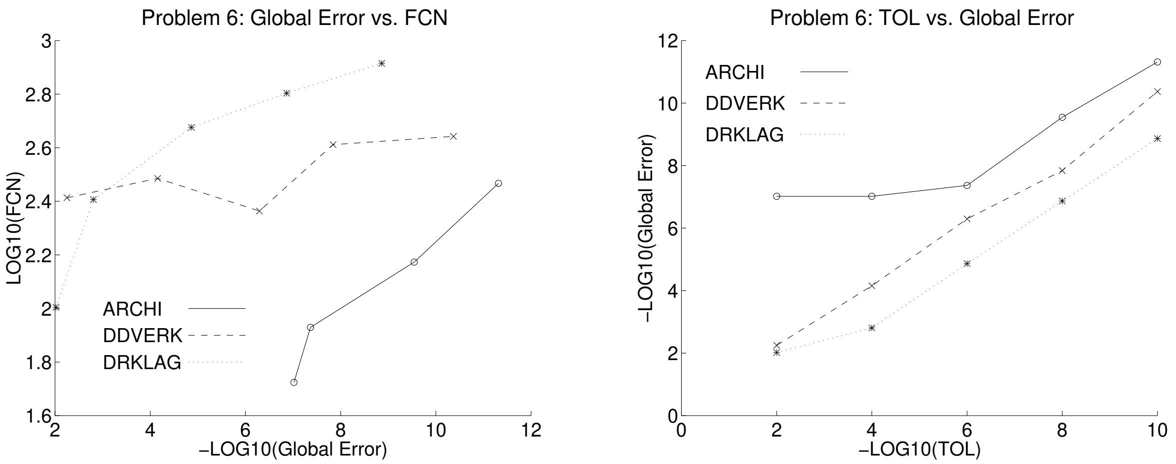

Figure 4.6: Global Error vs. FCN and TOL vs. Global Error Graphs for Problem 6

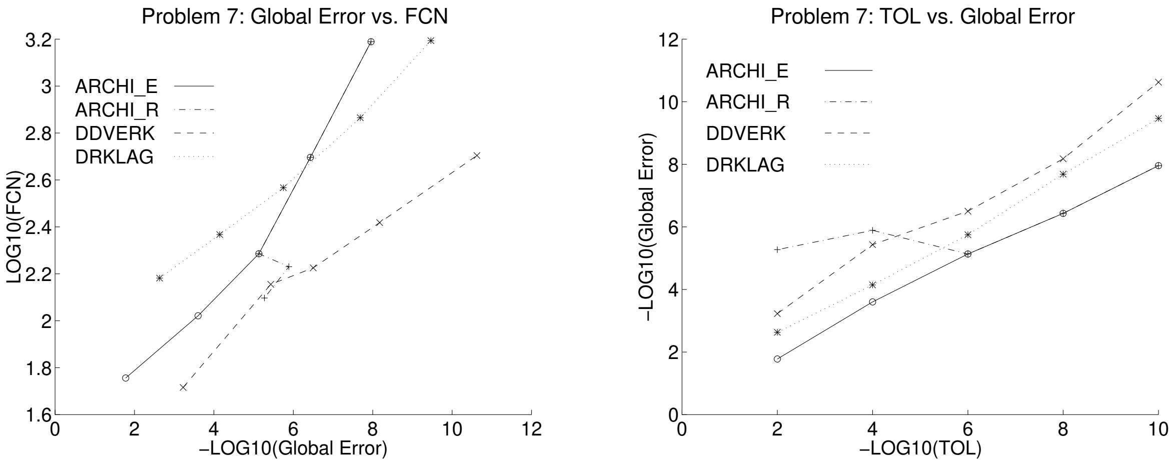

Figure 4.7: Global Error vs. FCN and TOL vs. Global Error Graphs for Problem 7

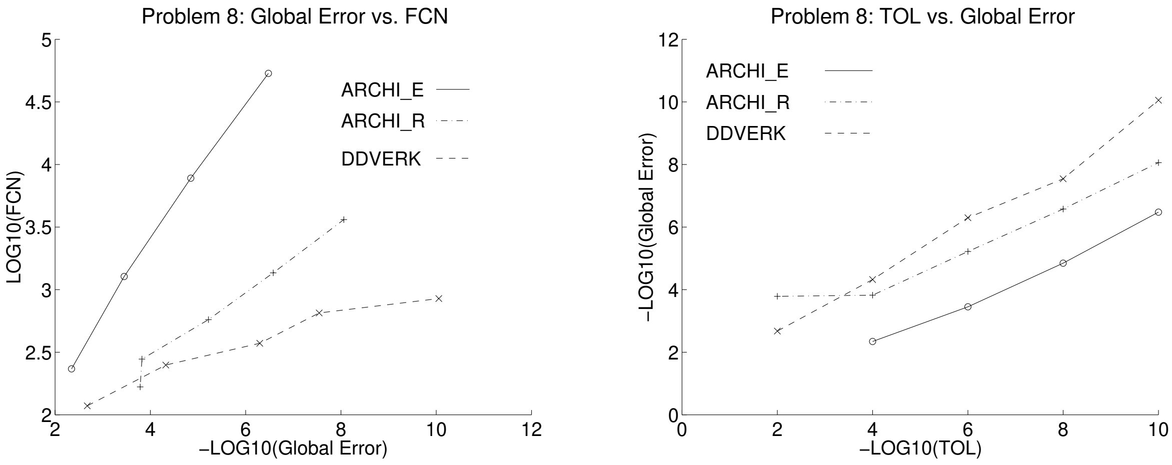

Figure 4.8: Global Error vs. FCN and TOL vs. Global Error Graphs for Problem 8 ($L_3 = 0.5$)

the cost for DDVERK is higher than ARCHI. One reason for this high cost is that the stepsize selection for DDVERK is strongly affected by the location of discontinuity points. Another reason is that the complexity to locate discontinuities for ARCHI is not related to the derivative evaluations while it is related to the derivative evaluations for DDVERK. It is also worth noting that ARCHI requires significantly more derivative evaluations for Problem 8, especially with stringent tolerances, than DDVERK.

It is observed from Table 4.11, defect is bounded by TOL for most problems. (For Problem 5, DEF1 is bounded by at most 300×TOL.) This result implies that a single point defect control mechanism works well and controls the defect over the whole interval.

The observed orders of convergence in Tables 4.12 and 4.13 for all problems except for Problem 6 are close to the expected order (6 for both global error and defect). For Problem 6, the stepsize is strongly affected by the location of (periodic) discontinuity points rather than accuracy requirement. There are four first order derivative discontinuities for Problem 6 and it took two steps on an average to locate each discontinuity with TOL$= 10^{-i}, i = 2, 4, 6$, three steps on an average with TOL$= 10^{-8}$, and five steps on an average with TOL$= 10^{-10}$. We also note that the (theoretical) order is derived by considering asymptotic behavior ($h \to 0$). For simple problems, the stepsize h may be too large for the asymptotic behavior to be observed. In addition, when severe tolerances are required and roundoff errors may not be negligible, observed order may not be consistent with the theoretical order.

Chapter 5

Conclusions

5.1 Summary

The main contribution of this thesis is the development and analysis of a numerical method for solving retarded and neutral type DDEs with vanishing delays. The method has been developed by adapting recently developed techniques for SDEs and developing a new approach to handle vanishing delays. The recently developed techniques include continuous Runge-Kutta formulas, an automatic handling technique for discontinuities, and defect control. The new approach is based on an iterative scheme determined by extrapolation or a special interpolant.

We determined convergence properties of the numerical solution. We first analyzed such properties for RDEs and then the analysis was extended to the case of NDEs. We showed that the global error associated with our class of methods should be bounded by a multiple of the prescribed tolerance for both RDEs and NDEs although order reductions may occur during some steps for NDEs. This is a similar situation to that which arises when using variable stepsize and variable order multistep methods for SDEs. Recall that for SDEs, a method usually starts with a low order formula and then increases the order according to the problem and tolerance. The theory behind their high orders of convergence for variable stepsize and variable order multistep methods is discussed in (Shampine and Zhang 1990). The proportionality of global error to tolerance is a desirable property and it was demonstrated in numerical experiments presented in Chapter 4 that three methods considered here have this property for most problems.

The analysis presented in Chapter 3 is analogous to 'backward error analysis' used often in linear algebra. The continuous approximation, $z(t)$, associated with our method satisfies the perturbed differential equation

$$z'(t) = f(t, z(t), z(t - \sigma(t, y(t))) + \delta(t).$$

We gave bounds for the perturbation $\|\delta(t)\|$ in §3.1.2. We then derived bounds for the effect of the perturbation $\delta(t)$ on the global error in §3.1.3.

We have implemented our method as an experimental Fortran code based on DVERK. We have carried out numerical experiments over various kinds of DDEs, reported the statistics, and compared the performance with ARCHI and DRKLAG in terms of cost and reliability.

A key advantage of DDVERK over ARCHI and DRKLAG lies in the automatic handling of discontinuities and small/vanishing delays. For ARCHI, the user is required to specify the dependency of propagated discontinuities among components and the option for small/vanishing delays. For DRKLAG, the user must specify if he/she wishes to use the rootfinding option or an extrapolation option for handling these difficulties. Another advantage is that DDVERK is the least expensive method for vanishing delay problems, as shown by the numerical results for Problems 3, 4, 7, and 8.

5.2 Future Work

It is important to improve the initial interpolant described in Algorithms 1 and 2 in §2.3 in order to make numerical solution for RDEs converge more rapidly when a small delay appears. This is considered by Enright and Hu (Enright and Hu 1994) where an interpolant, instead of extrapolation in Algorithm 1, is constructed using information from both the previous step and the current step. It is also important to improve the special interpolant in Algorithm 2 for NDEs so as to increase the theoretical order of convergence and to improve the efficiency.

Much of the analysis carried out in this thesis may be applied to the numerical solution of Volterra integro-differential equations although more complicated mathematical analysis may be needed. In addition, the effects of quadrature methods must be considered when approximating the solution of Volterra integro-differential equations.

Multipoint defect sampling for error control may improve the reliability of the method. Defect control based on 2-point sampling is applied to boundary value problems (BVP) of ordinary differential equations by Enright and Muir (Enright and Muir 1993). An improvement of reliability over 1-point sampling (without significantly increasing the cost) is demonstrated by their numerical results. Recall that BVPs usually require more function evaluations per step compared with SDEs and DDEs. Thus, extra function evaluations for defect control are usually small compared to the total costs for BVPs.

We solved only a standard set of problems and presented observations based on numerical results for this set of problems. The reliability and robustness for more difficult problems (especially large systems of equations with multiple delays) is one of the areas for future investigations.

Comparisons among ARCHI, DDVERK, and DRKLAG were provided in Chapter 4. More detailed comparisons (using a testing package such as DETEST and NDETEST (Enright and Pryce 1987) for SDEs) of softwares for DDEs is also a subject for future investigations.

Appendix A

Prologue of DDVERK

```
      SUBROUTINE DDVERK(N,ND,NW,IND,LOLD,T,TEND,Y,DY,TOL,TOLD,KOLD,W,C,
     +                  FCN,SIGMA,YINIT,DYINIT)


C***********************************************************************
C                                                                      *
C          Description of DDVERK (Experimental Code)                   *
C                                                                      *
C     DDVERK is a driver which invokes delay and discontinuity routines.  *
C     This driver first initializes parameters and then calls the DVERK  *
C     integration routine, RDMETH. After each step of the integration,  *
C     RDMETH returns to this driver and this driver checks if a        *
C     discontinuity exists in the step. If the existence of a          *
C     discontinuity is suspected, this driver calls DSLOC to find the  *
C     discontinuity point. Finally, if the step is accepted, DDVERK    *
C     calls STORE or STORED to store information necessary to construct  *
C     an interpolant in the future.                                    *
C                                                                      *
C***********************************************************************
C                                                                      *
C                         Calling sequence                             *
C                                                                      *
C     CALL DDVERK(N,ND,NW,IND,LOLD,T,TEND,Y,DY,TOL,TOLD,KOLD,W,C,      *
C    +            FCN,SIGMA,YINIT,DYINIT)                              *
C                                                                      *
C     INTEGER N,ND,NW,IND,LOLD                                         *
C     DOUBLE PRECISION T,TEND,Y(N),DY(N),TOL,TOLD(LOLD),KOLD(NW,7,LOLD)  *
C     DOUBLE PRECISION W(NW,20+20*ND),C(31)                            *
C     EXTERNAL FCN,SIGMA,YINIT,DYINIT                                  *
C                                                                      *
C***********************************************************************
C                                                                      *
```

```
C                        Arguments for DDERK                            *
C                                                                       *
C    N       Number of equations                                        *
C    ND      Number of different delay arguments                        *
C    NW      First dimension of workspace W and queue KOLD              *
C            NW must be at least MAX(N,ND)                              *
C    IND     Indicator --- same as DVERK except for IND = 7:            *
C            IND is set to be 7 after a discontinuity is found.         *
C    LOLD    Length of queue                                            *
C    T       Independent value                                          *
C    TEND    Value of T to which integration is to be carried out       *
C    Y       Solution vector at T                                       *
C    DY      Derivative vector of solution at T                         *
C    TOL     Tolerance                                                  *
C    TOLD    Queue of independent values T                              *
C    KOLD    Queue of derivative and solution approximations            *
C            The first dimension of KOLD must be NW                     *
C            The second dimension of KOLD must be at least 7            *
C            The third dimension of KOLD must be LOLD                   *
C    W       Workspace array                                            *
C            The first dimension of W must be NW                        *
C            The second dimension of W must be at least 20 + 2*ND       *
C    C       Communication vector - The dimension must be greater than  *
C            or equal to 31. C(1) to C(24) are the same as DVERK.       *
C    FCN     Name of subroutine for computing derivative                *
C            (must be supplied by a user)                               *
C    SIGMA   Name of subroutine for computing delay argument            *
C            (must be supplied by a user)                               *
C    YINIT   Name of subroutine for computing initial values on the     *
C            initial interval. (must be supplied by a user)             *
C    DYINIT  Name of subroutine for computing derivative of initial     *
C            values on the  initial interval. (must be supplied by      *
C            a user)                                                    *
C                                                                       *
C************************************************************************
C                                                                       *
C                           Communication vector                        *
C                                                                       *
C     C(25) Initial time                                                *
C     C(26) The first index of queue                                    *
C     C(27) The last index of queue                                     *
C     C(28) Flag indicating if K1 needs to be computed in the next      *
C           step: if C(28) = 0.0 then we need to compute K1 in the next *
C           step                                                        *
C     C(29) Flag indicating which of the extrapolation or the special   *
C           interpolant must be used: if C(29) = 0.0 then we use        *
C           extrapolation                                               *
C     C(30) TL values used in discontinuity locating routine DSLOC.     *
```

```
C     C(31) TH - TL used in DSLOC                                         *
C                                                                         *
C**************************************************************************
C                                                                         *
C                              Work array                                 *
C                                                                         *
C     W(*, 1:11)          K1 to K11 of CRK formula                        *
C     W(*, 12)            Solution at the right endpoint of the           *
C                         each step                                       *
C     W(*, 13)            Work space                                      *
C     W(*, 14)            Estimate of Defect                              *
C     W(*, 15)            Delay argument                                  *
C     W(*, 16)            Work space                                      *
C     W(*, 17)            Work space                                      *
C     W(*, 18)            Work space                                      *
C     W(*, 19)            Initial value (derivative)                      *
C     W(*, 20)            Initial value (solution)                        *
C     W(*, 21:20+ND)      Solution Delay values                           *
C     W(*, 21+ND:20+2*ND) Derivative Delay values                         *
C                                                                         *
C**************************************************************************
```

Appendix B

A Sample Program

```
C ... Sample program to solve Problem 7 by DDVERK

C ... Declaration of variables
C     .. Parameter integers ..
      INTEGER LOLD,N,NC,ND,NW
C        LOLD: the size of history queue
C        N: the number of equations
C        NC: the size of communication array (must be greater than 31)
C        ND: the number of different delay arguments
C        NW: the leading dimension of work array (must be greater than
C            max(N,ND) )

C     .. Parameters ..
      PARAMETER(LOLD=100,N=1,NC=31,ND=1,NW=1)

C     .. Scalar integers ..
      INTEGER I,IND
C        I: index for DO LOOP
C        IND: indicator

C     .. Scalar doubles ..
      DOUBLE PRECISION T,TEND,TOL
C        T: independent variable
C        TEND: end time of integration
C        TOL: tolerance

C     .. Array doubles ..
      DOUBLE PRECISION Y(N),DY(N),C(NC),W(NW,30)
      DOUBLE PRECISION TOLD(LOLD),KOLD(NW,7,LOLD)
C        Y: solution
C        DY: derivative of the solution
C        C: communication vector
C        W: work array (the second dimension of W should be more
C           than 20 + 2*ND.)
```

```
C        TOLD: vector for storing T values
C        KOLD: array for storing solutions and derivatives
C              (the second dimension must be 7)

C     .. External subroutines ..
      EXTERNAL SIGMA,FCN,YINIT,DYINIT

C ... Solve the problem over different tolerances from 10^(-2) to 10^(-10)

      DO 10 I = -2, -10, -2
         TOL = 10.D0**I
         PRINT '("************************************")'
         PRINT '("TOL =", 1PD9.2)', TOL

C ... Initialization
         IND = 1
         T = 0.D0
         Y(1) = 0.D0
         TEND = 1.4D0

C ... Loop
 30      CALL DDVERK(N,ND,NW,IND,LOLD,T,TEND,Y,DY,TOL,TOLD,KOLD,W,C,
     +               FCN,SIGMA,YINIT,DYINIT)

         IF (T .LT. TEND) GOTO 30
C ... End loop

C ... Print out the statistics
         PRINT '(/"IND = ", I3)', IND
         PRINT '("T = ", F12.5)', T
         PRINT '("Y = ", D26.15)', Y
         PRINT '("FCN = ", F10.0)', C(24)
         PRINT '("STEP = ", F10.0)', C(22)
         PRINT '("Global Error at T = ", D16.5)',
     +        DABS(Y(1) - DSIN(T))

 10   CONTINUE

      END



      SUBROUTINE FCN(N,ND,NW,T,Y,YDEL,DYDEL,DY)

C ... This subroutine computes derivative

C     ... Scalar arguments
      INTEGER N,NW,ND
```

```
      DOUBLE PRECISION T
C       N: the number of equations (input)
C       NW: the leading dimension of work array W (input)
C       ND: the number of different delay arguments (input)
C       T: independent variable (input)

C     .. Array arguments
      DOUBLE PRECISION Y(N),YDEL(NW,ND),DYDEL(NW,ND),DY(N)
C       Y: solution at T (input)
C       YDEL: i, j element contain i-th component of delay
C             value at j-th delay argument (input)
C       DYDEL: i, j element contain i-th component of
C              derivative delay value at j-th delay argument
C              (input)
C       DY: derivative value at T (output)

      DY(1) = DCOS(T) * (1.D0 + YDEL(1,1)) + Y(1) * DYDEL(1,1)
     +     - DSIN(T + T * DSIN(T)**2)

      RETURN
      END



      SUBROUTINE SIGMA(N,ND,T,Y,DELARG)

C ... This subroutine computes delay arguments

C     .. Scalar arguments ..
      INTEGER N,ND
      DOUBLE PRECISION T
C       N: the number of equations (input)
C       ND: the number of different delay argument (input)
C       T: independent variable (input)

C     .. Array arguments ..
      DOUBLE PRECISION Y(N),DELARG(ND)
C       Y: solution at T (input)
C       DELARG: delay argument (output)

      DELARG(1) = T * Y(1)**2

      RETURN
      END



      SUBROUTINE YINIT(N,T,Y)
```

```
C ... This subroutine computes solutions at T on initial interval

C     .. Scalar arguments ..
      INTEGER N
      DOUBLE PRECISION T
C       N: the number of equations (input)
C       T: independent variable (input)

C     .. Array arguments ..
      DOUBLE PRECISION , Y(N)
C       Y: initial value at T (output)

        Y(1) = 0.D0

      RETURN
      END



      SUBROUTINE DYINIT(N,T,DY)

C ... This subroutine computes derivatives at T on initial interval

C     .. Scalar arguments ..
      INTEGER N
      DOUBLE PRECISION T
C       N: the number of equations (input)
C       T: independent variable (input)

C     .. Array arguments ..
      DOUBLE PRECISION DY(N)
C       Y: initial derivative value at T (output)

      DY(1) = 1.D0

      RETURN
      END
```

References

Arndt, H. 1984. Numerical solution of retarded initial value problems: Local and global error and stepsize control. *Numer. Math.*, 43:343–360.

Baker, C. T. H. and Paul, C. A. H. 1993a. A global convergence theorem for a class of PCERK methods and vanishing lag DDEs. Private communication.

Baker, C. T. H. and Paul, C. A. H. 1993b. Parallel continuous Runge-Kutta methods and vanishing lag delay differential equations. *Advanced in Computational Mathematics*, 1:367–394.

Brauer, F. and Nohel, J. A. 1989. *The Qualitative Theory of Ordinary Differential Equations: An Introduction.* Dover, New York.

Castleton, R. N. and Grimm, L. J. 1973. A first order method for differential equations of neutral type. *Math. Comput.*, 27:571–577.

Dormand, J. R. and Prince, P. J. 1986. Runge-Kutta triples. *Computers Math. Applic.*, 12A(9):1007–1017.

Driver, R. D. 1977. *Ordinary and Delay Differential Equations.* Springer, Berlin.

El'sgol'ts, L. E. and Norkin, S. B. 1973. *Introduction to the Theory and Application of Differential Equations with Deviating Arguments.* Academic Press, New York.

Enright, W. H. 1989a. Analysis of error control strategies for continuous Runge-Kutta methods. *SIAM J. Numer. Anal.*, 26:588–599.

Enright, W. H. 1989b. A new error-control for initial value solvers. *Applied Maths. and Computation*, 31:288–301.

Enright, W. H. 1993. The relative efficiency of alternative defect control schemes for high order continuous Runge-Kutta formulas. *SIAM J. Numer. Anal.*, 30(5):1419–1445.

Enright, W. H. and Higham, D. J. 1991. Parallel defect control. *BIT*, 31:647–663.

Enright, W. H. and Hu, M. 1994. Interpolating Runge-Kutta methods for vanishing delay differential equations. Technical Report 292/94, Department of Computer Science, University of Toronto.

Enright, W. H., Jackson, K. R., Nørsett, S. P., and Thomsen, P. G. 1986. Interpolants for Runge-Kutta formulas. *ACM Trans. Math. Soft.*, 12:193–218.

Enright, W. H., Jackson, K. R., Nørsett, S. P., and Thomsen, P. G. 1988. Effective solution of discontinuous IVPs using a Runge-Kutta formula pair with interpolants. *Applied Maths. and Computation*, 27:313–335.

Enright, W. H. and Muir, P. 1993. A Runge-Kutta type boundary value ODE solver with defect control. Technical Report 267/93, Department of Computer Science, University of Toronto.

Enright, W. H. and Pryce, J. 1987. Two FORTRAN packages for assessing initial value methods. *ACM Trans. Math. Soft.*, 13(1):1–27.

Feldstein, A. and Neves, K. W. 1984. High order methods for state-dependent delay differential equations with nonsmooth solutions. *SIAM J. Numer. Anal.*, 21:844–863.

Gatica, J. and Waltman, P. 1982. A threshold model of antigen antibody dynamics with fading memory. In Lakshmikantham, editor, *Nonlinear phenomena in mathematical science*, pages 425–439, New York. Academic Press.

Gladwell, I., Shampine, L. F., Baca, L. S., and Brankin, R. W. 1987. Practical aspects of interpolation in use Runge-Kutta codes. *SIAM J. Sci. Stat. Comput.*, 8:322–341.

Grimm, L. J. 1971. Existence and continuous dependence for a class of nonlinear neutral differential equation. *Proc. Amer. Soc.*, 29(3).

Hairer, E., Nørsett, S. P., and Wanner, G. 1993. *Solving Ordinary Differential Equations I.* Springer-Verlag, New York, second edition.

Higham, D. J. 1989a. Defect estimation in Adams PECE codes. *SIAM J. Sci. Stat. Comput.*, 10:964–976.

Higham, D. J. 1989b. Highly continuous Runge-Kutta interpolants. Technical Report 220/89, Department of Computer Science, University of Toronto.

Higham, D. J. 1989c. Robust defect control with Runge-Kutta schemes. *SIAM J. Numer. Anal.*, 26:1175–1183.

Higham, D. J. 1989d. Runge-Kutta defect control using Hermite-Birkhoff interpolation. Technical Report 221/89, Department of Computer Science, University of Toronto.

Horn, M. K. 1983. Fourth- and fifth-order, scaled Runge-Kutta algorithms for treating dense output. *SIAM J. Numer. Anal.*, 20(3):558–568.

Hull, T. E., Enright, W. H., and Jackson, K. R. 1976. User's guide for DVERK–a subroutine for solving non-stiff ODE's. Technical Report 100, Department of Computer Science, University of Toronto.

Jackiewicz, Z. 1981a. The numerical solution of Volterra functional differential equations of neutral type. *SIAM J. Numer. Anal.*, 4:615–626.

Jackiewicz, Z. 1981b. One step methods for the numerical solution of Volterra functional differential equations of neutral type. *Applicable Anal.*, 12:1–11.

Jackiewicz, Z. 1982. Adams methods for neutral functional differential equations. *Numer. Math.*, 39:221–230.

Jackiewicz, Z. 1984. One-step methods of any order for neutral functional differential equations. *SIAM J. Numer. Anal.*, 21(3):486–511.

Jackiewicz, Z. 1986. Quasilinear multistep methods and variable step predictor-corrector methods for neutral functional differential equations. *SIAM J. Numer. Anal.*, 23(2):423–452.

Jackiewicz, Z. 1987. Variable-step variable-order algorithm for the numerical solution of neutral functional differential equations. *Appl. Numer. Math.*, 3:317–329.

Jackiewicz, Z. and Lo, E. 1991. The numerical solution of neutral functional differential equations by Adams predictor-corrector methods. *Appl. Numer. Math.*, 8:477–491.

Karoui, A. and Vaillancourt, R. 1994. Computer solutions of state-dependent delay differential equations. *Computers Math. Applic.*, 27(4):37–51.

Karoui, A. and Vaillancourt, R. 1995. A numerical method for vanishing-lag delay differential equations. *Appl. Numer. Math.*, 17:383–395.

Kuang, Y. 1991. On neutral delay logistics Gauss-type predator-prey systems. *Dyn. Stab. Systems*, 6:173–189.

Kuang, Y. 1993. *Delay Differential Equations with Applications in Population Dynamics.* Academic Press, Boston.

Kuang, Y. and Feldstein, A. 1991. Boundedness of solutions of a nonlinear nonautonomous neutral delay equation. *J. Math. Anal. Appl.*, 156:293–304.

Macdonald, N. 1989. *Biological Delay Systems: Linear Stability Theory.* Cambridge University Press, Cambridge.

Neves, K. W. 1975a. Algorithm 497, automatic integration of functional differential equations. *Collected Algorithm From ACM.*

Neves, K. W. 1975b. Automatic integration of functional differential equations: An approach. *ACM Trans. Math. Soft.*, 1:357–368.

Neves, K. W. 1981. Control of interpolatory error in retarded differential equations. *ACM Trans. Math. Soft.*, 7:421–444.

Neves, K. W. and Feldstein, A. 1976. Characterization of jump discontinuities for state dependent delay differential equations. *J. Math. Anal. Appl.*, 56:689–707.

Neves, K. W. and Thompson, S. 1992a. Software for the numerical solution of system of functional differential equations with state-dependent delays. *Appl. Numer. Math.*, 9:385–401.

Neves, K. W. and Thompson, S. 1992b. Solution of systems of functional differential equations with state dependent delays. Technical Report TR-92-003, Computer Science, Radford University.

Oberle, H. J. and Pesch, H. J. 1981. Numerical treatment of delay differential equations by Hermit interpolation. *Numer. Math.*, 37:235–255.

Oppelstrup, J. 1978. The RKFHB4 method for delay differential equations. In *Numerical treatment of differential equations, Lecture Notes in Mathematics 631*, pages 133–146, New York. Springer.

Paul, C. A. H. 1992a. Developing a delay differential equation solver. *Appl. Numer. Math.*, 9:403–414.

Paul, C. A. H. 1992b. *Runge-Kutta Methods for Functional Differential Equations.* PhD thesis, Manchester Univ., England.

Paul, C. A. H. 1994. A test set of functional differential equations. Technical Report 243, The Department of Mathematics, The University of Manchester, Manchester, England.

Shampine, L. F. 1985. Interpolation for Runge-Kutta methods. *SIAM J. Numer. Anal.*, 22:1014–1027.

Shampine, L. F. 1986. Some practical Runge-Kutta formulas. *Math. Comput.*, 46:135–150.

Shampine, L. F. and Zhang, W. 1990. Rate of convergence of multistep codes started by variation of order and stepsize. *SIAM J. Numer. Anal.*, 27(6):1506–1518.

Tavernini, L. 1971. One-step methods for the numerical solution of Volterra functional differential equations. *SIAM J. Numer. Anal.*, 8:786–795.

Verner, J. H. 1993. Differentiable interpolants for high-order Runge-Kutta methods. *SIAM J. Numer. Anal.*, 30(5):1446–1466.

Wheldon, T., Kirk, J., and Finlay, H. 1974. Cyclical granulopoiesis in chronic granulocytic leukemia: A simulation study. *Blood*, 43:379–387.

Willé, D. R. and Baker, C. T. H. 1992. The tracking of derivative discontinuities in systems of delay-differential equations. *Appl. Numer. Math.*, 9:209–222.