NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada
On the Numerical Solution of Delay Differential Equations

by

Abderrazek KAROU1

A thesis presented
to
the School of Graduate Studies
of the
University of Ottawa

in partial fulfilment of the requirements for the degree of
MASTER OF SCIENCE in MATHEMATICS

Department of Mathematics, University of Ottawa,
Ottawa, Ontario, Canada K1N 6N5

© Abderrazek Karoui, Ottawa, Canada, 1992
The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Abstract

A numerical method for the treatment of non-vanishing lag state dependent delay differential equations is developed in this work. This method is based on a (5,6) Runge–Kutta formula pair. The delayed term is approximated by a three-point Hermite polynomial. In order to obtain a highly accurate numerical scheme, special attention is given to the characterization and the localization of the derivative jump discontinuities of the solution. Some real-life problems are used to test the new method and compare it with existing ones.
Acknowledgements

Grateful thanks to Dr. Rémi Vaillancourt for his suggestions, guidance, and very constructive criticism during the course of my research. I would like also to thank Dr. P. W. Sharp for his advices, criticism and help he gave me.

A special thank to my friends in Canada and in Tunisia for their moral support.

I gratefully acknowledge the Agence Canadienne de Développement Internationale ACDI for providing me with financial support throughout my study in Canada.

My deepest appreciation go to my family for their moral support and most of all their love. I am specially grateful to my loving mother, Khadija, for all the care and love she gave me.
Contents

Abstract i

Acknowledgements ii

Table of contents iii

1 Introduction 1

2 Derivative Jump Discontinuities 5

2.1 Characterization of Jump Discontinuities for State Dependent DDE 5

2.1.1 Primary discontinuities ........................................... 5

2.1.2 Continuity class .................................................. 7

2.1.3 Examples .......................................................... 9

2.1.4 Generalized smoothing ........................................... 10
2.2 Influence of Discontinuities on the Numerical Solutions ............. 11

2.3 Numerical Methods to Locate Jump Discontinuities .................... 14

2.3.1 The Neves and Feldstein method ..................................... 14

2.3.2 The automatic inclusion of the discontinuities ....................... 15

2.3.3 The switching function method ...................................... 16

2.4 Numerical Results .................................................................. 21

3 Numerical Solution for Delay Equations .................................. 25

3.1 Runge-Kutta Formulae for OIVP ........................................... 25

3.2 Runge-Kutta Formulae for DDEs .......................................... 27

3.3 Error Bounds and Order of Convergence ................................ 29

3.3.1 General results .............................................................. 29

3.3.2 Local truncation error ..................................................... 34

3.3.3 Global truncation error ................................................... 38

4 Stepsize Control Policy for DDEs ............................................ 42

4.1 Stepsize Control using step doubling ..................................... 42
4.2 Stepsize Control Using a Pair of Formulae .................................. 44
4.3 Numerical Results ................................................................. 46
4.4 Comparison With Other Methods ............................................. 50

5 Conclusion ............................................................................. 53

Bibliography ............................................................................ 54

Appendix A .............................................................................. 58
Chapter 1

Introduction

Consider the state dependent delay differential system (DDEs):

\[ y'(t) = f(t, y(t), y(\alpha(t, y(t)))) \quad \text{for} \quad t \in [a, b], \]
\[ y(t) = \phi(t) \quad \text{for} \quad t \in [\bar{a}, a], \tag{1.1} \]

where

\[ y : [a, b] \to \mathbb{R}^n, \quad f : [a, b] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n, \]
\[ \alpha = (\alpha_1, \ldots, \alpha_n) : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n, \]
\[ \alpha_i(t, y(t)) \leq t \quad i = 1, \ldots, n. \]
\[ \bar{a} = \min\{\alpha(t, y(t)) ; \text{for} \ t \in [a, b]\} \text{ and } \phi : [\bar{a}, a] \to \mathbb{R}^n. \]

\[ y(\alpha(t, y(t))) \text{ is the delayed term and } t - \alpha(t, y(t)) \text{ is the lag.} \]

In recent years, there has been a growing interest in the numerical treatment of DDEs. This is due to the recurrence of such equations in various areas, including neural network theory, epidemiology, pharmacokinetics, and time lag control processes. Two examples of DDEs applications follow:
1. Population growth \cite{10} as a model of biological processes can be modeled by the state dependent DDE

\[ y'(t) = \frac{b(y(t)) - b(y(t - L(y(t))))}{1 - L'(y(t))b(y(t - L(y(t))))}, \]

where \( y(t) \) is the population size, and \( L(\cdot) \) is the lifespan of individuals in the population.

2. The variation in market price, \( p(t) \), of a particular commodity \cite{11} can be modeled by the following DDE

\[ p'(t) = p(t) f(D(p(t)), S(p(t - T))), \]

where \( D(\cdot) \) and \( S(\cdot) \), denote the demand and supply functions, respectively, for the commodity in question.

For more examples of applications of DDEs, the reader is referred to \cite{20}.

Two different procedures are conceivable for the numerical integration of (1.1). The first one, known as the method of steps, is attributed to Bellman \cite{21}; it approximates the delayed term \( y(\alpha(t, y(t))) \) by a repeated integration of the differential equation up to the abscissa \( \alpha(t, y(t)) \). Bellman's method has the advantage of low storage requirements and the disadvantage of increasing drastically the number of differential equations to be considered.

The procedure which is known as the extended ordinary differential equations (ODE) method uses a numerical integration scheme known from ODE together with an appropriate approximation formula to evaluate the delayed term. If a discrete
ODE method is used in the numerical integration, the approximation is given by an interpolation process. Otherwise, if the method itself provides an accurate continuous extension of the solution, interpolation is no longer needed. Several schemes based on ODE methods have been developed. Fourth-order Runge–Kutta methods and two-point Hermite interpolation have been used by Neves [14], and algorithms based on a fourth and seventh-order Runge–Kutta–Fehlberg methods together with Hermite interpolations are presented by H. J. Pesh and H. J. Oberle [9]. Recently Skip Thompson [23,24] has developed numerical methods which are based on a continuously imbedded Runge–Kutta method of Sarafyan. This method has the advantage that no extra function evaluations are needed to construct the interpolation polynomial. Finally, an algorithm based on a predictor-corrector mode of a one-step collocation method at $k$ Gaussian points has been made by Alfredo Bellen and Marino Zennaro [1].

In addition to the difficulties for ODEs, the numerical treatment of DDEs has to pay attention to two other difficulties. Firstly, the interpolation of the delayed term and secondly the jump discontinuities in the various derivatives of the solution.

In this work, the results of Neves [16] on the characterization of derivatives jump discontinuities is used and three numerical methods to locate them are presented in section 2, where special attention is paid to our method. The numerical integration method and its convergence properties are the subject of section 3. This method is based on a (5,6) Runge–Kutta formula pair to integrate the DDE and a three-point Hermite polynomial is used to interpolate the delayed term. In section 4, two different stepsize controls are presented; the method is tested numerically with respect to speed
and precision and is compared with other methods. The numerical experiments were
run in double precision on the AMDAHL 4370 at the University of Ottawa.

Finally, a non-vanishing lag is considered in this work. The functions $f$, $\Phi$ and $\alpha$
are assumed to be smooth enough and $\bar{a} < a$. 


Chapter 2

Derivative Jump Discontinuities

2.1 Characterization of Jump Discontinuities for State Dependent DDE

2.1.1 Primary discontinuities

For reason of simplicity, we shall consider only a scalar DDE, however the results of this section are valid for systems of equations with multiple lags. Hence, we consider the scalar delay differential equation

\[ y'(t) = f(t, y(t), y(\alpha(t, y(t)))) \quad \text{if } t \in [a, b], \]
\[ y(t) = \phi(t) \quad \text{if } t \in [\bar{a}, a], \]  

(2.1)

where \( \bar{a} = \min\{\alpha(t, y(t)) \mid t \in [a, b]\} \). The solution of (2.1) usually has an initial derivative jump discontinuity at the point \( t = a \). A set of derivative discontinuities, as it will be seen later, is propagated from the initial jump at \( a \). This set forms the so-called primary discontinuities of \( y(t) \).

Remark: The smoothness of \( f, \phi \) and \( \alpha \) does not necessarily imply that the solution
$y(t)$ is smooth. As an example, consider the DDE:

$$y'(t) = \frac{1}{t} y(t) \left( \frac{t \exp(y(t) - 2)}{y(t)} \right) \quad \text{if} \quad t \in [1.2.6].$$
$$y(t) = 1 \quad \text{if} \quad t \in [0.5.1]. \quad \text{(2.2)}$$

The explicit solution of (2.2) is as follows:

for $te^{y(t)-2}/y(t) \leq 1$,

$$y'(t) = \frac{y(t)}{t} \Rightarrow y(t) = t \quad \text{if} \quad 1 \leq t \leq 2;$$

for $2 \leq t \leq 2.6$,

$$y'(t) = \frac{1}{t} y(t) \frac{te^{y(t)-2}}{y(t)} \Rightarrow y(t) = 2 - \log(c - t).$$

where $c$ is some constant. Since $y(2) = 2$, then

$$y(t) = 2 - \log(3 - t).$$

Finally, the DDE (2.2) has the unique solution:

$$y(t) = \begin{cases} 1 & \text{if} \quad 0 \leq t \leq 1, \\ t & \text{if} \quad 1 \leq t \leq 2, \\ 2 - \log(3 - t) & \text{if} \quad 2 \leq t \leq 2.6. \end{cases}$$

Although $\phi$ is smooth enough in the interval $[0.5,1]$, $f$ and $\alpha$ are smooth in $[0.5,2.6]$ but the solution has a jump discontinuity in its second derivative at $t = 2$.

If $\xi$ is a point of discontinuity of either $f$ or $\phi$ then $\xi$ can generate a set of secondary discontinuities in the various derivatives of $y(t)$.

From now on, we assume that in (2.1) the functions $f$, $\phi$ and $\alpha$ are sufficiently smooth; hence we shall consider only the primary derivative discontinuities of $y(t)$.

Furthermore, the derivatives jump discontinuities are assumed to be isolated.
2.1.2 Continuity class

In this section, it will be seen that a necessary condition to obtain a high order numerical method for (2.1) is the accurate locating of a certain number of derivative jump discontinuities. The location of these jumps can be easily accomplished when the delay term \( \alpha(t, y(t)) \) is constant or a function of \( t \), but much more work is needed when the delay also depends on the solution \( y(t) \). The following discussion on the characterization of these discontinuities is based on the work of Alan Feldstein and Kenneth W. Neves [16, 17].

**Definition 1** Let \( Y \) be a real number and consider the switching function \( g(t) = \alpha(t, y(t)) - Y \). If \( \exists \ Z \in [a, b] \) such that \( Z \) satisfies

1. \( g(Z) = 0 \),
2. \( g^{(m)}(Z^-) \) exists from the left,
3. \( m \) is the least positive integer such that \( g^{(m)}(Z^-) \neq 0 \).

then \( Z \) is called a left zero of multiplicity \( m \) of \( g \).

**Definition 2** We say that problem 2.1 has continuity class \( p \) if the following hold over the appropriate domains:

1. \( f \) is continuous in its first argument and is uniformly Lipschitz continuous in its second and third arguments;
(2) for $i + j + k \leq p - 1$, all the mixed partial derivatives $f_{ijk}$ are continuous:

(3) $\phi \in C^{p-1}[a,a]$ and $\phi(a) = y(a)$:

(4) all the mixed partial derivatives $\alpha_{i,j}$ are continuous for all $i$ and $j$ such that $i + j \leq p$.

The main result of this section is contained in the following theorem which characterizes the possible derivative jump discontinuities of the solution of problem (2.1).

**Theorem 1** Assume that problem (2.1) has continuity class $p \geq 1$. Let $y(t)$ denote a solution of (2.1) and $Z_1 \in [a,b]$ be a discontinuity point of the $n^{th}$ derivative, $y^{(n)}(t)$, where $n \leq p$ and $n$ is the least integer such that $y^{(n)}$ is discontinuous at $Z_1$. Also assume that $y^{(p)} \in C[Z_1 - \xi_1, Z_1] \cup C[Z_1, Z_1 + \xi_1]$ for some positive real number $\xi_1$. If there exists $Z_2 \in [a,b]$ such that $Z_2$ is a zero of integer multiplicity $m \geq 1$ of $\alpha(t, y(t)) - Z_1$, then there exists $\xi_2 > 0$ such that

$$y^{(p)} \in C[Z_2 - \xi_2, Z_2] \cup C[Z_2, Z_2 + \xi_2] \quad \text{and} \quad y^{(q)} \in C[Z_2 - \xi_2, Z_2 + \xi_2],$$

where

(1) $q = p$ if $m$ is even,

(2) $q = \min\{p, m \cdot n\}$ if $m$ is odd.

**Proof:** See[17], page 846.

**Remark:** From the above theorem, one can see that the points of discontinuity are characterized as zeros of the nonlinear equation $g(t) = \alpha(t, y(t)) - Z$, where $Z$ is a
previous derivative discontinuity. Since, usually the first jump occurs at $Z = a$, a set of primary discontinuities is propagated from $a$.

2.1.3 Examples

Example 1: The following DDE:

$$
y'(t) = \frac{1}{t} y(t) y \left( \frac{t}{y(t)} e^{(y(t)-2)} \right) \quad \text{if} \quad t \in [1, 2.6],
$$

$$
y(t) = 1 \quad \text{if} \quad t \in [0.5, 1], \quad (2.3)
$$

has the exact solution:

$$
y(t) = \begin{cases} 
1 & \text{if} \quad t \in \left[ \frac{1}{2}, 1 \right], \\
t & \text{if} \quad t \in \left[ 1, 2 \right], \\
2 - \log(3 - t) & \text{if} \quad t \in \left[ 2, 2.6 \right].
\end{cases}
$$

Then, for $Z_1 = 1$ we have $y'(Z_1 - 0) = 0$ and $y'(Z_1 + 0) = 1$. Moreover the exact root of $\alpha(t, y(t)) - Z_1$ in this case is equal to 2 and it is clear that $y''(2 - 0) = 0$ and $y''(2 + 0) = -1$. This corresponds to the case where $n = 1, m = 1$ and $p$ is arbitrary in theorem 1.

Example 2: In [16], Neves and Feldstein presented the following DDE:

$$
y'(t) = y(t) + y \left( y(t) - (t - 2) - (t - 2)^2 \right) - \frac{1}{e} \quad \text{if} \quad t \geq 1,
$$

$$
y(t) = \frac{1}{e} \quad \text{if} \quad t \leq 1. \quad (2.4)
$$

The solution of (2.4) is

$$
y(t) = \begin{cases} 
\exp(t - 2) & \text{if} \quad t \in [1, 2], \\
B(t) & \text{if} \quad t \in [2, 2 + k],
\end{cases}
$$
where \( k \) is a positive real number and \( B(t) \) satisfies the ODE:

\[
B'(t) = B(t) + \exp \left[ B(t) - (t - 2) - \frac{1}{2}(t - 2)^2 - 2 \right] - \frac{1}{c},
\]

\[
B(2) = 1.
\]

Now set \( A(t) = B(t) - (t - 2) - \frac{1}{2}(t - 2)^2 - 2 \). Then

\[
B^{(2)}(t) = B^{(1)}(t) + A^{(1)}(t) \exp[A(t)],
\]

\[
B^{(3)}(t) = B^{(2)}(t) + \left[ A^{(3)}(t) + (A^{(1)}(t))^2 \right] \exp[A(t)],
\]

\[
B^{(4)}(t) = B^{(3)}(t) + \left[ A^{(3)}(t) + 3A^{(1)}(t)A^{(2)}(t) + (A^{(1)}(t))^3 \right] \exp[A(t)].
\]

Hence

\[
y^{(i)}(2 - 0) = y^{(i)}(2 + 0) = 1, \quad i = 1, 2, 3,
\]

and

\[
1 + c = y^{(4)}(2 + 0) \neq y^{(4)}(2 - 0) = 1.
\]

This corresponds to the case where \( Z_1 = 1, n = 1, m = 3 \) and \( p \) is arbitrary in theorem 1.

2.1.4 Generalized smoothing

The smoothing property of the solution \( y(t) \) of problem (2.1) is an important consequence of theorem 1. Let \( \Phi \) be \( J \)-incompatible (i.e. \( J \) is the least integer such that \( \phi^{(J)}(a - 0) \neq y^{(J)}(a + 0) \)), and define a sequence, \( S_k \), of point sets as follows:

\( S_0 = \{a\}, S_k = \bigcup \{Z; g(Z) = 0 \text{ with some odd multiplicity } m \leq p/J \} \), for \( k \leq p \),

where the union is taken over all \( Y \in S_{k-1} \) such that \( a \leq Y < Z < b \) and where
\[ g(t) = \alpha(t, y(t)) - Y. \] Now let
\[
M(Z) = \sup \{ q; y^{(q)} \text{ is continuous at } t = Z \},
\]
\[
N_k = \inf \{ M(Z); Z \in S_k \}.
\]

It is proved in [16] that if (2.1) has continuity class \( p \) and the initial function \( \phi(t) \) is \( J \)-incompatible, then
\[
N_{k+1} \geq N_k + 1.
\]

This result means that the smoothness of the solution \( y(t) \) increases by at least 1 as the index \( k \) (level of the discontinuity \( Z_k \)) increases by 1. This is the so-called generalized smoothing of \( y(t) \).

It is clear that the degree of continuity at jump points is not an increasing function of \( t \). If \( \alpha(t, y(t)) \) is sufficiently oscillatory, then a new discontinuity at level 1 occurs each time \( \alpha(t, y(t)) \) crosses the point \( a \); hence there may be an infinite number of level 1 discontinuities. If \( \alpha(t, y(t)) \) is a strictly increasing function of \( t \), then each set \( S_k \) is formed by a single point [16]. The generalized smoothing property of the solution \( y(t) \) is important in the sense that only a limited number of discontinuities have to be located to obtain a high order numerical method for problem (2.1).

### 2.2 Influence of Discontinuities on the Numerical Solutions

To proceed further, a brief description of the numerical method to solve the delay problem (2.1) is needed (the method will be described in more detail in the next
chapter) and we shall consider only one-step ODE methods which have been extended
to the delay problem. These DDE methods can be written as

\[ y_{n+1} = y_n + h \Psi(t_n, y_n, P_q(\alpha(t_n, y_n), y_n, y_n', h)). \]  

(2.5)

where

\[ \Psi \] is the increment function. For example

the \( r \)-stage Runge–Kutta method for problem (2.1) has the form

\[ y_{n+1} = y_n + h \sum_{i=1}^{r} c_i K_i. \]  

(2.6)

where

\[ K_i = f(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} K_j, P_q(\alpha(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} K_j))), \]

\[ i = 1, \ldots, r, \]

and \( P_q \) is an interpolation polynomial of degree \( q \) for the delay term

\( y(\alpha(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} K_j)). \) If

\( y_h(t) \) is the numerical solution of problem (2.1) obtained by applying the above:

Runge–Kutta method at \( t = t_n \).

\( Y \) is an exact discontinuity in the \( m^{th} \) derivative of \( y(t) \),

\( Y_h \) is an approximation to \( Y \),

\( Z \) is the exact zero of \( \alpha(t, y(t)) - Y \), and \( y^{(z)} \) is discontinuous at \( Z \),

\( Z_h \) is a numerical approximation to \( Z \),

\( \xi \) is a positive real number such that \( Y_h \) is the only discontinuity of \( y_h \) in \( [Y_h - \xi, Y_h + \xi] \),

12
the interpolation process is done by using the following rule:

for $t \in I_1 = [Y_h - \xi, Y_h]$, let the interpolation be formed from tabular points in $I_1$, and for $t \in I_2 = [Y_h, Y_h + \xi]$, the interpolation be formed from tabular points in $I_2$.

Then we have the following important theorem.

**Theorem 2** Assume the one-step method is of global order of convergence $p$ and let the interpolation process be done by using the above rule. If

$$|Y - Y_h| = O(h^n) \quad \text{for } n \geq 1,$$

$$|Z - Z_h| = O(h^r) \quad \text{for } r \geq 1,$$

then

$$|y(t) - y_h(t)| = O(h^q) \quad \forall \ t \in [Z_h, \eta]$$

for some positive real number $\eta$ and $q = \min\{p, n \cdot m, r \cdot z\}$

**Proof:** See [17], page 853.

**Remark:** From the previous theorem, it is seen that the accuracy with which discontinuities are located is critical to obtain a high order numerical method. The subject of the following section is the numerical approximation to these jump points.
2.3 Numerical Methods to Locate Jump Discontinuities

It is not possible to know a priori the exact derivative jump discontinuities $\xi_i$ of the solution unless the delay term is constant; hence a numerical approximation to these $\xi_i$ is needed. For this task, some methods have been developed [15,17,23]; we shall consider three of them in this work.

2.3.1 The Neves and Feldstein method

In [17], the following technique is suggested:

Divide the interval $[a,b]$ in $N$ subintervals of equal length, $h$, and let $t_j, j = 0, \ldots, n,$ be the grid points. Then use a one-step method adapted to DDE to compute a numerical approximation $y_h(t)$ to the solution. Since, in practice, there are a finite number of jump points of computational concern (i.e. isolated discontinuities), we order them by $a < Z_1 < \ldots < Z_m \leq b$ where $Z_j$ satisfies $Z_k = \alpha(Z_j, y(Z_j))$ and $Z_j$ is the ancestor of $Z_k$. Assume that $(i-1)$ derivatives jump discontinuities are located and we want to locate the $i^{th}$ jump; then the algorithm can be described as follows:

1. After each integration step, evaluate the switching function $\alpha(t, y_h(t)) - Z_{h,k}$ where $Z_{h,k}$, for some $k < i$, is an ancestor to the approximate jump $Z_{h,i}$. If $|\alpha(t_j, y_h(t_j)) - Z_{h, k}| \times |\alpha(t_{j+1}, y_h(t_{j+1}) - Z_{h,k}) > 0$, $[t_j, t_{j+1}]$ does not contain a (primary) discontinuity; hence advance the computation by another step.

But if the product is negative, then
2. Construct a Runge–Kutta polynomial interpolant of the appropriate degree; then by using a standard rootfinder find the zero of $\alpha(t, y_h(t)) - Z_{h,k}$, and take this root as the approximation to the jump point $Z_{h,i}$.

Remark: Since $Z_{h,i}$ is a zero of $\alpha(t, y_h(t)) - Z_{h,k} = 0$, where $y_h(t)$ is the interpolated solution in the interval containing a discontinuity, the interpolation process over an interval which contains a discontinuity leads usually to a loss of accuracy and consequently to an inaccurate localization of the root $Z_{h,i}$. This loss of accuracy also occurs in the next method, but not in the third method.

2.3.2 The automatic inclusion of the discontinuities

This second method, which is described in [14], is based on the idea that the two-point Hermite extrapolated solution, for sufficiently small $h$ ($h$ is the maximum step size between data points), is of the same order of accuracy as the two-point Hermite interpolated solution, provided that

1. $h_{n+1}/h_n$ is bounded,

2. there is no jump discontinuity in the interval $[t_n, t_{n+1}]$.

Furthermore, if the above two conditions are satisfied, then the two estimates should disagree only in $O(h^4)$ terms. Also, note that if the discrepancy between the interpolated and the extrapolated values of the solution is large, then we are in the presence
of a discontinuity. In this case the step size should be reduced to force the integrator to hit the jump.

2.3.3 The switching function method

To describe this method, we need to review briefly the Newton backward interpolation polynomial. Assume that the values of a function $f$ are known at $(n + 1)$ past, not necessarily equidistant, points, $t_N, t_{N-1}, \ldots, t_{N-n}$. An approximation to $f(t)$ at a point $t$, is given by the Newton backward divided difference formula:

$$f(t) = f(t_N) + (t - t_N)f[t_N, t_{N-1}] + (t - t_N)(t - t_{N-1})f[t_N, t_{N-1}, t_{N-2}] + \cdots + (t - t_N)(t - t_{N-1})\cdots(t - t_{N-n+1})f[t_N, \ldots, t_{N-n}] + E(t)$$

where $f[t_N, \ldots, t_{N-i}]$, denotes the $i^{th}$ divided difference of $f$. The error term of the formula is

$$E(t) = (t - t_N)\cdots(t - t_{N-n})f[t_N, \ldots, t_{N-n}, t].$$

Note that if $I = (t_{N-n}, \max\{t_N, t\})$ and $f \in C^{n+1}(I)$, then there exists $\xi \in I$ such that

$$f[t_N, \ldots, t_{N-n}, t] = \frac{1}{(n + 1)!}f^{(n+1)}(\xi).$$

For this approximation to be accurate, $t$ must be close to the end point $t_N$. 

16
Description of the switching function method:

From section (2.1), it is seen that a derivative jump discontinuity is a zero of the expression $\alpha(t, y(t)) - Y$. Then after each integration step, $[t_n, t_{n+1}]$, check whether the product $|\alpha(t_n, y_n) - Y_h| \times |\alpha(t_{n+1}, y_{n+1}) - Y_h| \leq 0$ or not where,

$t_{n+1}$ is the most recent grid point,

$y_{n+1}$ is a numerical approximation to $y(t_{n+1})$,

$Y_h$ is a numerical approximation to an exact previous derivative discontinuity.

The algorithm of the method is as follows:

1. If $|\alpha(t_n, y_n) - Y_h| \times |\alpha(t_{n+1}, y_{n+1}) - Y_h| > 0$, then proceed to the next integration step.

2. If $|\alpha(t_n, y_n) - Y_h| \times |\alpha(t_{n+1}, y_{n+1}) - Y_h| \leq 0$, then there is a derivative jump discontinuity in $[t_n, t_{n+1}]$. To locate this jump,

2.1 Construct the Newton backward extrapolation polynomial of $\alpha(t, y_h(t)) - Y_h$ from a suitable number of previous values of $\alpha(t_i, y_i) - Y_h$, $i = n - m, \ldots, n$.

2.2 Use the bisection method to find the zero $Z_h$ of this polynomial; then $Z_h$ is taken as an approximation to the exact root $Z$ of $\alpha(t, y(t)) - Y$.

2.3 Include $Z_h$ in the grid points so that $t_{n+1} = Z_h$.

The following important question arises: how accurate is the above method? The answer is given by the following theorem. For simplicity we assume a constant step-
size. However the result of the theorem is still valid for a system of equations with a variable stepsize \( h \).

**Theorem 3** Consider the DDE:

\[
y'(t) = f(t, y(t), y(\alpha(t, y(t)))) \quad \text{if} \quad t \in [a, b],
\]

\[
y(t) = \phi(t) \quad \text{if} \quad t \in [\bar{a}, b].
\]

(2.7)

Let

\[
g(t) = \alpha(t, y(t)) - Y, \quad g_h(t) = \alpha(t, y_h(t)) - Y_h,
\]

where \( y_h(t) \) is a numerical approximation to the solution \( y(t) \) of the DDE, \( Y \) and \( Y_h \) are the exact and the approximate derivative discontinuity of \( y(t) \), respectively. Assume that the function \( \alpha(t, X) \) is Lipschitzian with respect to \( X \) and \( g(t) \) has a zero in \((t_j, t_{j+1})\). Then, by using a Newton backward extrapolation polynomial \( P_{g_h(t)} \) of \( g_h(t) \) in \((t_j, t_{j+1})\), one can approximate the zero \( Z \) of \( g(t) \) by the zero \( Z_p \) of \( P_{g_h(t)} \) to the order \( O\left(h^{\min\{p/m,p/n\}}\right) \) provided that

1. the degree of the extrapolation polynomial is at least \((p - 1)\),

2. \(|Y - Y_h| = O(h^p)|,

3. the global integration method is at least of order \( p \).

4. \( Z^*_p \) and \( Z_p \), the nearest roots to \( Z \) of \( P_g \) and \( P_{g_h} \) are of multiplicity \( m \) and \( n \) respectively, where \( P_g \) is the Newton extrapolation polynomial of \( g \) on \((t_j, t_{j+1})\).

5. the divided difference \( g[t_j, \ldots, t_{j-p+1}, Z] \) is bounded.
Proof:

By theorem 1, we may assume that \( g(t) \) changes sign in \((t_j, t_{j+1})\); then for sufficiently small \( h \), \( g_h(t) \) also changes sign in the same interval (see [15]). Let \( P'_g(t) \) and \( P_{g_h}(t) \) be the Newton backward extrapolation polynomials of \( g(t) \) and \( g_h(t) \), respectively. These polynomials are constructed from the \( p \) grid points \( t_{j-p+1}, \ldots, t_j \). Assume that \( t_{j-p+1} > Y \), then

\[
P'_g(t) - P_{g_h}(t) = g(t_j) - g_h(t_j) + (t - t_j) \{ g[t_j, t_{j-1}] - g_h[t_j, t_{j-1}] \} + \cdots + (t - t_j) \cdots (t - t_{j-p+2}) \{ g[t_j, \ldots, t_{j-p+1}] - g_h[t_j, \ldots, t_{j-p+1}] \}.
\]

By induction on the positive integer \( k \), we prove

\[
(t - t_{j-m}) \cdots (t - t_{j-k-m}) \{ g[t_j-m, \ldots, t_{j-k-m-1}] - g_h[t_j-m, \ldots, t_{j-k-m-1}] \} = O(h^p),
\]

\( m = 0, \ldots, p - 2 - k \). \hspace{1cm} (*)

For \( k = 0 \),

\[
|g(t_j-m) - g_h(t_j-m)| = |\alpha(t_j-m, y(t_j-m)) - Y - \alpha(t_j-m, y_h(t_j-m)) + Y_h|
\leq L^\alpha |y(t_j-m) - y_h(t_j-m)| + |Y - Y_h|
\leq O(h^p) \hspace{1cm} \forall m = 0, \ldots, p - 1.
\]

Now assume that \((*)\) is true for \( k \leq p - 3 \); then for \( m = 0, \ldots, p - 1 - (k + 1) \), we have

\[
(t - t_{j-m}) \cdots (t - t_{j-m-k-1}) \{ g[t_{j-m}, \ldots, t_{j-m-k-2}] - g_h[t_{j-m}, \ldots, t_{j-m-k-2}] \}
= \frac{t - t_{j-m-k-1}}{t_{j-m} - t_{j-k-m-2}} (t - t_{j-m}) \cdots (t - t_{j-k-m}) \times
\times \{ g[t_{j-m}, \ldots, t_{j-k-m-1}] - g_h[t_{j-m}, \ldots, t_{j-k-m-1}] \} + \frac{t - t_{j-m}}{t_{j-m} - t_{j-k-m-2}} (t - t_{j-m-1}) \cdots (t - t_{j-k-m-2}) \times
\times \{ g_h[t_{j-m-1}, \ldots, t_{j-k-m-2}] - g[t_{j-m-1}, \ldots, t_{j-k-m-2}] \}.
\]
Since there exists a constant $M$ such that
\[
\frac{t - t_{j-m-k+1}}{t_{j-m} - t_{j-k-m-2}} \leq M \quad \text{and} \quad \frac{t - t_{j-m}}{t_{j-m} - t_{j-k-m-2}} \leq M.
\]
then using the induction assumption (*), it follows that
\[
(t - t_{j-m}) \cdots (t - t_{j-m-k+1}) [g[t_{j-m}, \ldots, t_{j-k-m-2}] - g_h[t_{j-m}, \ldots, t_{j-k-m-2}]] = O(h^p), \quad m = 0, \ldots, p - 2 - (k + 1).
\]
Hence (*) is also true for $k \leq p - 2$, and therefore
\[
|P_g(t) - P_{g_h}(t)| = O(h^p) \quad \forall t \in (t_j, t_{j+1}) \quad (**).
\]
Now let $Z$ be the zero of $g(t)$ in $(t_j, t_{j+1})$ and $Z_p^*$, $Z_p$ be the zeros nearest to $Z$ of $P_g(t)$ and $P_{g_h}(t)$, respectively. Then the error term is
\[
E(Z) = P_g(Z) - g(Z),
\]
\[
= (Z - t_j) \cdots (Z - t_{j-p+1}) g[t_j, \ldots, t_{j-p+1}, Z].
\]
Since there exists $M'$ such that $g[t_j, \ldots, t_{j-p+1}, Z] \leq M'$, then it is clear that
\[
|P_g(Z) - g(Z)| = O(h^p);
\]
therefore $P_g(Z) = O(h^p)$ because $g(Z) = 0$.

Also, a Taylor expansion of $P_g$ in a neighborhood of $Z_p^*$ gives
\[
P_g(Z) = P_g(Z_p^*) + \left[ \sum_{k=0}^{p-2} \frac{(Z - Z_p^*)^{k+1}}{(k + 1)!} P_g^{(k+1)}(Z_p^*) \right].
\]
As $Z_p^*$ is a zero of multiplicity $m$ of $P_g$, then $P_g^{(m)}(Z_p^*) \neq 0$ and since
\[
\left[ \sum_{k=1}^{p-2} \frac{(Z - Z_p^*)^{k+1}}{(k + 1)!} P_g^{(k+1)}(Z_p^*) \right]
\]
is bounded, then from the fact that $P_g(Z_p^*) = 0$ and $P_g(Z)$ is of order $O(h^p)$, one can conclude that
\[
|Z_p^* - Z| = O(h^{p/m}). \quad \text{(2.8)}
\]
Since \( P_s(Z_s^*) = 0 \), then (**) applied to \( Z_s^* \) gives

\[
P_{gh}(Z_s^*) = O(h^p).
\]

By a Taylor expansion of \( P_{gh}(Z_s^*) \) in a neighbourhood of \( Z_p \) and since \( Z_p \) is a zero of \( P_{gh} \) of integer multiplicity \( n \), then \( P_{gh}^{(n)}(Z_p) \neq 0 \) and since

\[
\left[ \sum_{k=0}^{p-2} \frac{(Z_p^* - Z_p)^{k+1}}{(k+1)!} P_{gh}^{(k+1)}(Z_p^*) \right]
\]

is bounded, one concludes that

\[
|Z_p^* - Z_p| = O(h^{p/n}). \quad \Box \tag{2.9}
\]

Now from (2.8), (2.9) and the inequality

\[
|Z - Z_p| \leq |Z_p^* - Z_p| + |Z_p^* - Z|,
\]

we obtain the desired result

\[
|Z - Z_p| = O\left(h^{\min(p/m,p/n)}\right).
\]

### 2.4 Numerical Results

The following numerical results illustrate the accuracy with which the switching function method locates the derivative jump discontinuities of the solution. The first two examples use a Runge–Kutta method of order 4 to solve the given DDE. An extrapolation polynomial of degree 3 is used to extrapolate the switching function, then the bisection method is used to locate to the machine precision (16 digits) the root of this polynomial. The computations in these two examples were made with a constant
Table 2.1: Error in the roots $Z_h$ for example 1

| Step size $h$ | $|Z - Z_h|$ |
|--------------|------------|
| 0.03         | 5.9E-10    |
| 0.06         | 1.2E-09    |
| 0.09         | 4.8E-08    |
| 0.12         | 1.7E-07    |

step size.

Example 1.

Consider the DDE:

$$y'(t) = \frac{1}{2\sqrt{2}} y(y(t) - \sqrt{2} + 1) \quad \text{if} \quad t \in [1,3],$$

$$y(t) = 1 \quad \text{if} \quad t \in [0,1],$$

(2.10)

which has a second jump discontinuity at $Z = 2$. The numerical results for different step sizes are given in table 2.1:

Example 2.

$$y'(t) = \frac{1}{t} y(t) y\left(\frac{te^{-y(t)-2}}{y(t)}\right) \quad \text{if} \quad t \in [1,4],$$

$$y(t) = 1 \quad \text{if} \quad t \in [0,1],$$

(2.11)

which has a second jump discontinuity at $Z = 2.0$. The numerical results for different step sizes are given in table 2.2.

In the next two examples, the Runge–Kutta–Verner (5,6) pair together with a three-point Hermite interpolation polynomial are used to integrate the DDEs and a Newton backward extrapolation polynomial of degree 5 is used to locate the first three jump derivative discontinuities. The implementation of this pair is described in the next chapter; here, we are only interested in the numerical approximation of the
Table 2.2: Error in the roots $Z_h$ for example 1

| Step size $h$ | $|Z - Z_h|$ |
|---------------|------------|
| 0.03          | 2.9E-10    |
| 0.06          | 2.8E-09    |
| 0.09          | 1.6E-08    |
| 0.12          | 1.7E-07    |

Table 2.3: Numerical results for example 3

| TOL   | $|R_1 - e|$ | $|R_2 - e^2|$ | $|R_3 - e_3|$ |
|-------|------------|--------------|--------------|
| 1E-05 | 1.10E-07   | 2.72E-07     | 4.32E-05     |
| 1E-07 | 5.08E-10   | 1.23E-08     | 1.14E-07     |
| 1E-09 | 1.26E-10   | 2.13E-10     | 7.29E-10     |
| 1E-12 | 3.77E-15   | 9.01E-14     | 8.52E-13     |

derivatives jump discontinuities. A variable step-size is adopted and the results are obtained by using different values of the tolerance.

**Example 3.**

\[
y'(t) = \frac{1}{t} y(t) y\left(\frac{t e^{y(t)-2}}{y(t)}\right) \quad \text{if} \quad t \in [1, 4], \\
y(t) = 1 \quad \text{if} \quad t \in [0, 1],
\]  \quad (2.12)

The first three jumps occur at $e, e^2, e_3 = \exp(3 - \exp(1 - e))$. Let $R_1, R_2, R_3$ denote the numerical approximation to $e, e^2, e_3$, respectively. Table 2.3 shows the absolute error made in locating these discontinuities.

**Example 4.**

\[
y'(t) = \frac{t - 1}{t} y(t - \log(t) - 1) \quad \text{for} \quad t \geq 1, \\
y(t) = 1 \quad \text{for} \quad t \in [0, 1].
\]  \quad (2.13)

Since the exact solution of (2.9) is not known and consequently the derivatives dis-
Table 2.4: Numerical results for example 4

| TOL | $|R_1 - \xi_1|$ | $|R_2 - \xi_2|$ |
|-----|-----------------|-----------------|
| 1E-05 | $1.49E - 06$    | $2.99E - 06$    |
| 1E-07 | $3.11E - 09$    | $3.79E - 09$    |
| 1E-09 | $1.00E - 10$    | $1.00E - 09$    |
| 1E-12 | $1.68E - 13$    | $1.29E - 13$    |

continuities cannot be found explicitly, reference values of the first two jumps as given in [8] are:

$$\xi_1 = 3.146 193 220 620 582 585 2, \quad \xi_2 = 5.925 449 824 508 246 492 6.$$  

Let $R_1$ and $R_2$ denote the numerical approximations to $\xi_1$ and $\xi_2$, respectively. Then the absolute error made in locating these jumps is given in table 2.4.

Conclusion

Despite the accuracy with which the switching function method approximates the derivatives jump discontinuities, it has the disadvantage of being difficult to implement in a routine to solve a system of delay equations. The number of switching functions to be handled in this case increases drastically with the size of the system and the number of possible derivatives jump discontinuities. In such situation, it is much more convenient to use the automatic inclusion of the discontinuities as described earlier.
Chapter 3

Numerical Solution for Delay Equations

It is well known that one-step methods for ordinary initial-value problems, as well as linear multistep methods, can be adapted to solve the delay problem (2.1). Since our numerical method, as it will be seen later, is based on a Runge–Kutta formula, a brief review of these formulae is given in section 3.1. In section 3.2, it is showed how to adapt a one-step method for ordinary initial value problems (OIVP) to delay problems. Finally, the convergence property of the numerical method is studied in section 3.3.

3.1 Runge–Kutta Formulae for OIVP

Consider the n-dimensional OIVP system:

\[ y'(t) = f(t, y(t)), \]
\[ y(t_0) = y_0. \] (3.1)

where \( f : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n \) and \( y \in \mathbb{R}^n \). An explicit \( r \)-stage Runge-Kutta method for problem (3.1) has the following form if it advances the solution from \( t_n \) to \( t_{n+1} \):

\[ y_{n+1} = y_n + h \Phi(t_n, y_n, h). \] (3.2)

where

\[
\Phi(t_n, y_n, h) = \sum_{i=1}^{r} c_i K_i,
\]

\[
K_i = (K_i^1, \ldots, K_i^n)^T.
\]

\[
K_i^k = f_k\left(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} K_j\right), \quad k = 1, \ldots, n.
\]

Most methods have

\[
\alpha_i = \sum_{j=1}^{i-1} \beta_{ij}, \quad i = 1, \ldots, r.
\]

Note that an \( r \)-stage Runge-Kutta method involves \( r \) function evaluations per step.

Each of the functions \( K_i \) may be interpreted as an approximation to the derivative \( y'(t) \) and \( \Phi(t, y, h) \) as a weighted mean of these approximations.

If one can choose \( \alpha_i, \beta_{ij} \) and \( c_i \) in such a way that the expansion of \( \Phi(t, y, h) \) differs from the Taylor expansion of the function \( f \) in a neighbourhood of \((t, y)\) only in the \( p^{\text{th}} \) and higher powers of \( h \), then the method is said to be of order \( p \). In 1968, Fehlberg [4] made an important contribution to Runge-Kutta methods by constructing classes of pairs, denoted by RKF\((p-1, p)\):

\[
y_{n+1} = y_n + h \sum_{i=1}^{r} c_i K_i, \quad \hat{y}_{n+1} = \hat{y}_n + h \sum_{i=1}^{s} \hat{c}_i K_i
\]

of order \( p-1 \) and \( p \), respectively, for \( p=6,7,8,9 \). Formula pairs use the same \( f \)-values, \( K_i \), to compute \( y_{n+1} \) to the order \( (p-1) \) with some few extra \( K_i \), to compute \( \hat{y}_{n+1} \).
Table 3.1: Butcher tableau for RK formula pair

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>α₂</td>
<td>β₂₁</td>
</tr>
<tr>
<td>α₃</td>
<td>β₃₁ β₃₂</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>αᵣ</td>
<td>βᵣ₁ βᵣ₂ ... βᵣᵣ₋₁</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>αᵣ</td>
<td>βᵣ₁ βᵣ₂ ... ... βᵣᵣ₋₁</td>
</tr>
<tr>
<td>c₁</td>
<td>c₂ ... cᵣ</td>
</tr>
<tr>
<td>ᵅc₁</td>
<td>ᵅc₂ ... ᵅcᵣ</td>
</tr>
</tbody>
</table>

to the order \( p \). A formula pair provides us, as it will be seen later, with a way to estimate the local truncation error of the numerical solution.

Finally, it is common to present a formula pair by the Butcher tableau shown in table 3.1. The Runge–Kutta–Verner (5, 6) pair, on which our numerical method is based, uses eight derivative evaluations per step and it is represented in table 3.2.

### 3.2 Runge–Kutta Formulae for DDEs

In order to use a Runge–Kutta formula to solve the \( n \)-dimensional DDEs:

\[
y'(t) = f(t, y(t), y(\alpha(t, y(t)))) \quad \text{if} \quad t \in [a, b],
\]

\[
y(t) = \phi(t) \quad \text{if} \quad t \in [\bar{a}, a],
\]

(3.3)

one has to redefine (3.2) as follows:

\[
y_{n+1} = y_n + h\Psi(t_n, y_n, \Psi(\alpha(t_n, y_n), y_i, y'_i), h)
\]

(3.4)

where
Table 3.2: RKV (5,6) pair

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
</tr>
<tr>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
</tr>
<tr>
<td>4/15</td>
<td>1/75</td>
<td>16/75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/3</td>
<td>5/6</td>
<td>8/3</td>
<td>5/2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5/6</td>
<td>-165/64</td>
<td>55/6</td>
<td>-425/64</td>
<td>85/96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>12/5</td>
<td>-8</td>
<td>4015/612</td>
<td>-11/36</td>
<td>88/255</td>
<td></td>
</tr>
<tr>
<td>1/15</td>
<td>-8263/15000</td>
<td>124/75</td>
<td>643/680</td>
<td>-81/250</td>
<td>2484/10625</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3501/1720</td>
<td>-300/43</td>
<td>297275/52632</td>
<td>-319/2322</td>
<td>24068/84065</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>12/180</th>
<th>0</th>
<th>2275/5984</th>
<th>5/16</th>
<th>12/85</th>
<th>1/44</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/40</td>
<td>0</td>
<td>875/2244</td>
<td>23/72</td>
<td>264/1955</td>
<td>0</td>
<td>125/11592</td>
<td>47/616</td>
</tr>
</tbody>
</table>
\[ \Psi(t_n, y_n, P_q(\alpha(t_n, y_n), y_i, y'_i, h)) = \sum_{i=1}^r c_i K_i, \]

\[ K_i = (K_i^1, \ldots, K_i^n)^T, \]

\[ K_i^k = f_k \left( t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} K_j, P_q(\alpha(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} K_j)) \right), \]

\[ k = 1, \ldots, n, \]

\( P_q \) is an interpolation polynomial of degree \( q \) used to approximate the delay term.

The interpolation polynomial is needed to evaluate the delay term which generally lies between two previous grid points. In our scheme \( P_q \) is taken to be a three-point Hermite interpolation polynomial. The reason for taking a fifth-degree interpolation polynomial together with a (5,6) Runge–Kutta formula pair will be seen in the following section. Finally, we shall refer to (3.4) as the adapted Runge–Kutta formula and to \( \Psi(t, y, P_q(\alpha(t, y), y_i, y'_i, h)) \) as the increment function corresponding to (3.4).

### 3.3 Error Bounds and Order of Convergence

#### 3.3.1 General results

Most of the material in this subsection is taken from [6]. Consider the general case where (3.1) has to be solved by a general \( k \)-step method of the form:

\[ \sum_{i=0}^k \alpha_i y_{n+i} = h \phi(t_n, y_{n+k}, \ldots, y_n, h). \]  

(3.5)

It is convenient to define the characteristic polynomial \( \rho(\theta) \) of (3.5) by

\[ \rho(\theta) = \sum_{i=0}^k \alpha_i \theta^i. \]
Note that one-step methods, including Runge-Kutta ones, are special cases of (3.5) where \( k = 1 \).

**Definition 3** The method (3.5) is convergent if, when applied to problem (3.3),
\[
y_h(t) \to y(t) \quad \text{as} \quad h \to 0
\]
for any \( t \in [a, b] \).

One can define in a similar way the convergence of an adapted ODE method for solving DDEs.

The following important question arises, what conditions the method (3.5) must satisfy in order to be convergent? To answer this question, the following definitions and results are needed.

**Definition 4** The local discretization error \( d_n \) of (3.5) at \( t_n \) is defined to be
\[
d_n = \frac{1}{\rho'(1)h} \left[ \sum_{i=0}^{k} \alpha_i y(t_{n+i}) - h \phi(t_n, y(t_n), \ldots, y(t_n), h) \right],
\]
where \( y(t) \) is the solution of (3.1). In particular, the local discretization error associated with (3.2) has the form:
\[
d_n = \frac{1}{h} |y(t_{n+1}) - y(t_n) - h \phi(t_n, y(t_n), h)|.
\]

We now state the minimal level of local accuracy of (3.5).

**Definition 5** The method (3.5) is said to be consistent if
\[
\max_{0 \leq n \leq N} \|d_n\| \to 0 \quad \text{as} \quad h \to 0,
\]
where \( N \) is the total number of steps. It is consistent of order \( p \) if

\[
\max_{0 \leq n \leq N} \|d_n\| = O(h^p).
\]

It is necessary to make a further definition related to the stability properties of (3.5) at this stage.

**Definition 6** The method (3.5) is said to satisfy the root condition if the roots of the characteristic polynomial \( \rho(\theta) \) lie within or on the unit circle, those on the unit circle being simple.

In particular, any consistent one-step method of the form (3.5) necessarily satisfies the root condition since the only root of \( \rho(\theta) \) is \( \theta_1 = 1 \).

We now state the following fundamental theorem of Dahlquist that has occupied a central position in the development of the numerical analysis of differential equations.

**Theorem 4** The method (3.5) is convergent if and only if it is consistent and satisfies the root condition.

In particular, the following theorem holds for any one-step method of the form (3.5).

**Theorem 5** A one-step method of the form (3.5) is convergent if and only if it is consistent.

Unlike the OIVP case, the study of the convergence property of the adapted Runge–Kutta scheme is complicated by the fact that an interpolation process is used.
in this scheme. Therefore some lemmas and theorems are needed to find the order of convergence of an adapted Runge–Kutta method. We start the study by proving the following fundamental lemma on the Lipschitz property of the increment function $\Psi$.

**Lemma 1** Assume that

1. $f$ is Lipschitzian with respect to its second and third arguments, with Lipschitz constants $M_2$ and $M_3$, respectively.

2. $\alpha(t, y)$ is Lipschitzian with respect to $y$, with Lipschitz constants $M_\alpha$.

3. $P_q(\alpha(t, y_n), y_i, y'_i)$ is Lipschitzian with respect to $\alpha$ and $y_i$ with Lipschitz constants $M_P$ and $M$ respectively.

Then there exists $L_\Psi$ such that

$$||\Psi(t, y_n, P_q(\alpha(t, y_n), y_i, y'_i), h) - \Psi(t, \tilde{y}_n, P_q(\alpha(t, \tilde{y}_n), \tilde{y}_i, \tilde{y}'_i), h)|| \leq L_\Psi \max_{i \leq n}||y_i - \tilde{y}_i||.$$  

**Proof:** Since

$$\Psi(t, y, P_q(\alpha(t, y), y_i, \tilde{y}_i), h) = \sum_{i=1}^{r} c_i K_i(y)$$

where

$$K_i(y) = f \left( t + \alpha_i h, y + h \sum_{j=1}^{i-1} \beta_{ij} K_j(y), P_q(\alpha(t + \alpha_i h, y + h \sum_{j=1}^{i-1} \beta_{ij} K_j(y), y_i, \tilde{y}_i) \right),$$

it suffices to prove that $\forall i = 1, \ldots, r$, there exists $L_i$ such that

$$||K_i(y_n) - K_i(\tilde{y}_n)|| \leq \max_{i \leq n}||y_i - \tilde{y}_i||.$$
This can be done by induction on \( i \). For \( i = 1 \) we have

\[
K_1(y_n) = f(t, y_n, P_q(\alpha(t, y_n), y_i, y'_i)),
\]

\[
K_1(\tilde{y}_n) = f(t, \tilde{y}_n, P_q(\alpha(t, \tilde{y}_n), t\tilde{y}_i, \tilde{y}'_i)).
\]

and

\[
\|K_1(y_n) - K_1(\tilde{y}_n)\| = \|f(t, y_n, P_q(\alpha(t, y_n), y_i, y'_i)) - f(t, \tilde{y}_n, P_q(\alpha(t, \tilde{y}_n), \tilde{y}_i, \tilde{y}'_i))\| \leq M_2\|y_n - \tilde{y}_n\| + M_3 \left[ M_p M_\alpha \|y_n - \tilde{y}_n\| + M \|y_i - \tilde{y}_i\| \right] \leq L_1 \max_{i \leq n} \|y_i - \tilde{y}_i\|.
\]

Now assume that the result holds for \( m < r \). Thus \( \exists L_m \) such that

\[
\|K_m(y_n) - K_m(\tilde{y}_n)\| \leq L_m \max_{i \leq n} \|y_i - \tilde{y}_i\|.
\]

Then

\[
\|K_{m+1}(y_n) - K_{m+1}(\tilde{y}_n)\| = \|f(t + \alpha_{m+1} h, y + h \sum_{j=1}^{m} \beta_{m+1,j} K_j(y_n), P_q(\alpha(t + \alpha_{m+1} h, y_n + h \sum_{j=1}^{m} \beta_{m+1,j} K_j(y_n)), y_i, y'_i)) - f(t + \alpha_{m+1} h, \tilde{y}_n + h \sum_{j=1}^{m} \beta_{m+1,j} K_j(\tilde{y}_n), P_q(\alpha(t + \alpha_{m+1} h, \tilde{y}_n + h \sum_{j=1}^{m} \beta_{m+1,j} K_j(\tilde{y}_n)), \tilde{y}_i, \tilde{y}'_i))\| \leq M_2\|y_n - \tilde{y}_n\| + M_2 h \sum_{j=1}^{m} |\beta_{m+1,j}| L_j \max_{i \leq n} \|y_i - \tilde{y}_i\| + M_3 \left\{ M_p M_\alpha \times \right. \left[ \|y_n - \tilde{y}_n\| + h \sum_{j=1}^{m} L_j |\beta_{m+1,j}| \max_{i \leq n} \|y_i - \tilde{y}_i\| \right] + M \max_{i \leq n} \|y_i - \tilde{y}_i\| \right\} \leq \left[ M_2 + M_3 M_p M_\alpha + M_3 M + h (M_2 + M_3 M_p M_\alpha) \sum_{j=1}^{m} L_j |\beta_{m+1,j}| \right] \times \max_{i \leq n} \|y_i - \tilde{y}_i\| \leq L_{m+1} \max_{i \leq n} \|y_i - \tilde{y}_i\|.
\]
Hence the result is also true for $m + 1$. As $\Psi$ is a linear combination of the $K_i$, then there exists a positive constant $L_\Psi$ such that

$$
||\Psi(t, y_n, P_q(\alpha(t, y_n), y_i, y'_i) \cdot h) - \Psi(t, \tilde{y}_n, P_q(\alpha(t, \tilde{y}_n), \tilde{y}_i, \tilde{y}'_i) \cdot h)|| \\
\leq L_\Psi \max_{i \leq n} ||y_i - \tilde{y}_i||. \quad \Box
$$

**Remarks:**

1. By using the above induction technique, one can prove that the increment function corresponding to any Runge–Kutta formula satisfies the Lipschitz condition in $y$ provided that $f$ in (3.1) is Lipschitzian with respect to $y$.

2. The Lipschitz property of the increment function $\Psi$ is important because, as it will be seen later, it provides us with an elegant way to have a bound for the global truncation error of a numerical method by using only a bound on the local truncation. This will be the subject of the next subsection.

### 3.3.2 Local truncation error

To find an upper bound for the local truncation error of an adapted Runge–Kutta scheme, a local version of problem (2.1) is needed; hence we assume that the exact solution $y(t)$ of (2.1) is known up to $t = t_n$ and consider the following problem on the interval $[t_n, t_{n+1}]:$

$$
\tilde{y}'(t) = f(t, \tilde{y}(t), P_q(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i))), \\
\tilde{y}(t_n) = y(t_n),
$$

where $P_q(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i))$ is the interpolation polynomial of $\tilde{y}(\alpha(t, \tilde{y}(t)))$ constructed by using previous values of $y$ and $y'$ at some nodes $t_i$.

It is known that the local truncation error of a $p^{th}$ order RK method for an OIVP is of the order $(p + 1)$ on the interval $[a, b]$ provided that the solution $y(t) \in C^p[a, b]$. 

34
For an adapted \( p \)th order RK method, we have the following claim.

**Claim 1** The local truncation error of an adapted \( p \)th order Runge-Kutta method is of order \((p+1)\) with respect to problem (3.6) provided that:

1. \( y(t) \in C^p[t_n, t_{n+1}] \),

2. the interpolation polynomial is smooth enough in \([t_n, t_{n+1}]\),

i.e. the set of interpolation points is fixed during the integration step.

**Proof:**

Since \( \tilde{y}(t), P_q(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i)) \in C^p[t_n, t_{n+1}] \), then problem (3.6) is equivalent to

\[
\begin{align*}
\tilde{y}(t) & = F(t, \tilde{y}(t)), \\
\tilde{y}(t) & = y(t_n),
\end{align*}
\]

with

\[
F(t, \tilde{y}(t)) = f(t; \tilde{y}(t), P_q(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i))), \quad t \in [t_n, t_{n+1}].
\]

Hence (3.7) is an OIVP that satisfies the required differentiability conditions on \( F \) and \( \tilde{y}(t) \). Then the local truncation error associated with the numerical solution of (3.6) that is obtained by a \( p \)th order RK method is of the order \( O(h^{p+1}) \) where \( h \) is the current stepsize. Since the numerical solution \( \tilde{y}_h(t) \) of (3.6) obtained by an adapted Runge-Kutta method coincides with that of (3.7), one concludes that the local truncation error with respect to problem (3.7) is of the order \( O(h^{p+1}) \). \( \square \)
It is clear that (3.6) is a modification of the original problem (1.1) and consequently the corresponding local truncation errors are not necessarily equal. If \( y(t_{n+1}) \) and \( y_{n+1} \) are the exact and the numerical solutions of (2.1), respectively, then the following important question arises: what can be said about the local error \( ||y(t_{n+1}) - y_{n+1}|| \)?

The answer is given in the following lemma.

**Lemma 2** Assume that

1. \( y \) is Lipschitzian,
2. \( ||P_q(\alpha(t, \bar{y}(t), y(t_i), y'(t_i)) - y(\alpha(t, \bar{y}(t)))|| \leq L h^\gamma, \)
   where \( L \) is a positive constant and \( t \in [t_n, t_{n+1}] \),
3. the conditions of claim1 and lemma1 hold.

Then

\[
||y(t_{n+1}) - y_{n+1}|| \leq M h^{\min\{p,q\}+1} \text{ for some positive constant } M.
\]

The proof of this lemma requires the following version of Gronwall's inequality:

**Theorem 6 (Gronwall's inequality)**

Let \( h(t) \geq 0, \phi(t) \geq 0 \) be real continuous functions on \([a, b]\) and \( A \) be a fixed constant. If

\[
0 \leq h(t) \leq A + \int_a^t \phi(s) h(s) \, ds, \quad t \in [a, b],
\]

36
\[ 0 \leq h(t) \leq A \exp \left( \int_a^t \phi(s) \, ds \right). \]

**Proof of lemma 2:** From \( y(t_n) = y_n \) and the equalities
\[
y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t), y(\alpha(t, y(t)))) \, dt,
\]
\[
\tilde{y}_{n+1} - \tilde{y}_n = \int_{t_n}^{t_{n+1}} f(t, \tilde{y}(t), P_\tau(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i))) \, dt,
\]
on one obtains
\[
\|y(t_{n+1}) - \tilde{y}_{n+1}\| = \int_{t_n}^{t_{n+1}} \| f(t, y(t), y(\alpha(t, y(t)))) - f(t, \tilde{y}(t), P_\tau(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i))) \| \, dt,
\]
\[
\|y(t_{n+1}) - \tilde{y}_{n+1}\| \leq \int_{t_n}^{t_{n+1}} \left[ L^2 \|y(t) - \tilde{y}(t)\| + L^2 \|y(\alpha(t, y(t))) - P_\tau(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i))\| \right] \, dt.
\]

Since
\[
\|y(\alpha(t, y(t))) - P_\tau(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i))\| \\
\leq \|y(\alpha(t, y(t))) - y(\alpha(t, \tilde{y}(t)))\| \\
+ \|y(\alpha(t, \tilde{y}(t))) - P_\tau(\alpha(t, \tilde{y}(t)), y(t_i), y'(t_i))\|,
\]
by condition (1), one obtains
\[
\|y(\alpha(t, y(t))) - y(\alpha(t, \tilde{y}(t)))\| \leq L' \|y(\alpha(t, y(t))) - \alpha(t, \tilde{y}(t))\| \\
\leq L' L^a \|y(t) - \tilde{y}(t)\|.
\]

Now, we use condition (2) of lemma 1 to obtain
\[
\|y(t_{n+1}) - \tilde{y}_{n+1}\| \leq \int_{t_n}^{t_{n+1}} \left[ M_2 \|y(t) - \tilde{y}(t)\| \\
+ M_3 M_\alpha \|y(t) - \tilde{y}(t)\| \right] \, dt + M_3 L h^q \\
\leq M_3 L h^{q+1} + \int_{t_n}^{t_{n+1}} (M_2 + M_3 M_\alpha) \|y(t) - \tilde{y}(t)\| \, dt.
\]
Next, Gronwall's inequality with $e(t) = ||y(t) - \tilde{y}(t)||$ gives us

$$e(t_{n+1}) \leq LM_3 h^{q+1} \exp \int_{t_n}^{t_{n+1}} (M_2 + M_3 M_y M_\alpha) \, dt \leq LM_3 h^{q+1} e^{(M_2 + M_3 M_y M_\alpha) h}.$$  \hspace{1cm} (3.8)

By the above claim, $\exists M'$ such that

$$||y_{n+1} - \tilde{y}_{n+1}|| \leq M' h^{p+1}.$$  \hspace{1cm} (3.9)

Finally, from (3.8), (3.9) and the triangle inequality we get

$$||y(t_{n+1}) - y_{n+1}|| \leq ||y(t_{n+1}) - \tilde{y}_{n+1}|| + ||\tilde{y}_{n+1} - y_{n+1}|| \leq M'' h^{q+1} + M' h^{p+1} \leq M h^{\min\{p,q\}+1}. \quad \Box$$

**Remark:** From the inequality $||y(t_{n+1}) - y_{n+1}|| \leq M'' h^{q+1} + M' h^{p+1}$ it can be seen that the local truncation of the method involves two types of errors. The first one is of the order $O(h^{q+1})$ and corresponds to the error due by the $q^{th}$ degree interpolation polynomial, the second is of the order $O(h^{p+1})$ and corresponds to the local truncation error of the $p^{th}$ order integration method. Hence a $(p - 1)^{st}$ degree interpolation polynomial is needed together with a $p^{th}$ integration method so that the numerical method has a local truncation error of the order $O(h^{p+1})$. This explains our choice of the fifth degree Hermite interpolation polynomial and the Runge–Kutta–Verner (5,6) pair in our scheme.

### 3.3.3 Global truncation error

The last part of this section is concerned with the global error of the numerical method. The following theorem gives us a bound for this global error.
Theorem 7 Let $y(t)$ and $y_h(t)$ denote the exact and the numerical solution of problem (2.1), respectively. If

1. $y_h$ is obtained by a stable numerical method for ordinary differential equations together with an interpolation scheme,

2. the local error of this method is of the order $p + 1$,

then the global error $||y(t) - y_h(t)||$ over the interval $[a, b]$ is of the order $p$.

Proof:

If $e_n = y_n - y(t_n)$ denotes the global truncation error of the numerical method, then

$$y_{n+1} = y_n + h\Psi(t_n, y_n, P_q(\alpha(t_n, y_n), y_i, y_i'), h),$$

and

$$e_{n+1} = y_{n+1} - y(t_{n+1})$$

$$= y_n + h\Psi(t_n, y_n, P_q(\alpha(t_n, y_n), y_i, y_i'), h) - y(t_{n+1})$$

$$\Rightarrow e_{n+1} = e_n + h[\Psi(t_n, y(t_n), P_q(\alpha(t_n, y(t_n)), y(t_i), y'(t_i)), h)]$$

$$- \Psi(t_n, y(t_n), P_q(\alpha(t_n, y(t_n)), y(t_i), y'(t_i)), h)$$

$$+ h[\Psi(t_n, y(t_n), P_q(\alpha(t_n, y(t_n)), y(t_i), y'(t_i)))]$$

$$- [y(t_{n+1}) - y(t_n)]$$

$$= e_n + h[\Psi(t_n, y(t_n), P_q(\alpha(t_n, y(t_n)), y_i, y_i'), h) - \Psi(t_n, y(t_n), P_q(\alpha(t_n, y(t_n)), y(t_i), y'(t_i)), h) + R_n(h).$$

(3.10)

where

$$R_n(h) = y(t_{n+1}) - y_n - h\Psi(t_n, y_n, P_q(\alpha(t_n, y(t_n)), y(t_i), y'(t_i)), h).$$

39
Hence, by lemma 2,

\[ R_n(h) = O(h^{p+1}). \]  \hfill (3.11)

Also, since the interpolation polynomial \( P_q \) satisfies the condition (3) of lemma 1, then there exists a positive constant \( L_\psi \) such that

\[ \| \Psi(t_n, y_n, P_q(\alpha(t_n, y_n, h), y_i, y_i')) - \Psi(t_n, y(t_n), P_q(\alpha(t_n, y(t_n), y(t_i), y'(t_i))) \| \leq L_\psi \max_{i \leq n} \| y_i - y(t_i) \|. \]

Hence from (3.9), (3.10) and (3.11) one obtains the following estimation for the global discretization error of the form

\[ \| e_{n+1} \| \leq (1 + hL\psi) \max_{i \leq n} \| e_i \| + Kh^{p+1}. \]  \hfill (3.12)

Now let \( D = Kh^{p+1} \) and \( L = L_\psi \). Then by induction on \( n \) we prove that

\[ \| e_n \| \leq D \frac{(1 + hL)^n - 1}{hL} + (1 + hL)^n \| e_0 \|. \]  \hfill (3.13)

This inequality is trivial for \( n = 0 \). Assume that it is true \( \forall i \leq n \). From (3.12), we have

\[ \| e_{n+1} \| \leq \| 1 + hL \| \max_{i \leq n} \| e_i \| + D. \]

Since

\[ \| e_i \| \leq D \frac{(1 + hL)^i - 1}{hL} + (1 + hL)^i \| e_0 \|, \quad i \leq n \]

it follows that

\[ \| e_{n+1} \| \leq (1 + hL)D \frac{(1 + hL)^n - 1}{hL} + (1 + hL)^n \| e_0 \| + D \]

\[ \leq D \frac{(1 + hL)^{n+1} - 1}{hL} + (1 + hL)^{n+1} \| e_0 \|. \]
Hence (3.13) is also true for $n + 1$, and therefore it is true for any positive integer $n$. If $n$ denote the total number of steps taken over the interval $[a, b]$, then $nh \leq (b - a)$. Since $1 + hL \leq e^{hL}$, it is easy to see that

$$
||e_n|| \leq D \frac{(e^{hL})^n - 1}{hL} + (e^{hL})^n ||e_0||,
$$

$$
||e_n|| \leq D \frac{e^{L(b-a)} - 1}{hL} + e^{L(b-a)} ||e_0||.
$$

This proves that the global discretization error is of the order $p$ because $D = O(h^{p+1})$ and $||e_0|| = 0$ in the initial value problem. □

**Summary**

In this section, we have shown that the adapted $p^{th}$ order Runge–Kutta method is consistent since the local discretization error of the method is of the order $O(h^{\min(p,q)})$. Hence, the method is convergent. Furthermore, a more powerful result has been shown, that is, under some additional conditions on the degree of the interpolation polynomial, the order of the basic method and the approximation of the derivative jump discontinuities, the global truncation error of the adapted method is of the order $O(h^p)$. 

41
Chapter 4

Stepsze Control Policy for DDEs

4.1 Stepsize Control using step doubling

The local truncation error $T_{n+1}$ at $t_{n+1}$ of the general explicit one-step method for an OIVP is defined to be

$$T_{n+1} = y(t_{n+1}) - y(t_n) - h \Phi(t_n, y(t_n), h),$$

where $y$ is the exact solution of the nonlinear OIVP and $\Phi$ is the increment function of the method. In particular for the $p^{th}$ order Runge–Kutta methods, the local truncation error has the form

$$T_{n+1} = \Psi(t_n, y(t_n))h^{p+1} + O(h^{p+2}),$$
where $\Psi(t, y(t))$ is the principal error function and $\Psi(t_n, y(t_n))$ is the principal local truncation error. If it is assumed that no previous errors have been made, then

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

To apply the step doubling technique, we have to compute a second approximation $y_{n+1}^*$ to $y(t_{n+1})$ by applying the same method at the grid point $t_{n-1}$ with a step length of $2h$. Therefore, we have the following expression:

$$y(t_{n+1}) - y_{n+1}^* = \Psi(t_{n-1}, y(t_{n-1}))(2h)^{p+1} + O(h^{p+2}).$$

By expanding $\Psi(t_{n-1}, y(t_{n-1}))$ about $\Psi(t_n, y(t_n))$, one gets

$$y(t_{n+1}) - y_{n+1}^* = \Psi(t_n, y(t_n))(2h)^{p+1} + O(h^{p+2}).$$

This implies that

$$y_{n+1} - y_{n+1}^* = (2^{p+1} - 1)\Psi(t_n, y(t_n))h^{p+1} + O(h^{p+2}).$$

Hence an estimation of the local truncation error may be written as:

$$\Psi(t_n, y(t_n))h^{p+1} = \frac{y_{n+1} - y_{n+1}^*}{2^{p+1} - 1}.$$ 

Thus to apply the step doubling, we have to compute the numerical solution over two successive steps using step length $h$ and then recompute over the double step using step length $2h$. The difference between the two values of $y$ divided by $(2^{p+1} - 1)$ is an estimation of the local truncation error for OIVPs as well as for delay problems.
The above estimation rule has been widely used for the estimation of the local truncation error for delay problems [7,14,23]. Since, the step doubling technique requires an increase of 50% in the computational effort, an alternative cheaper way to estimate the local truncation error of the numerical method is needed. This is the subject of the next section.

4.2 Stepsize Control Using a Pair of Formulae

The use of a Runge–Kutta formula pair is an elegant way to estimate the local truncation error of a numerical method. Instead of using one Runge–Kutta method of order \( p \), a basis method of order \( (p+1) \) is used to advance the numerical computation and a control method of order \( p \) to estimate the local truncation error. In our scheme \( p \) is taken to be 5. Now if \( \hat{y}_h(t_n) \) and \( y_h(t_n) \) denote the numerical approximations of the \( (p + 1) \)th and \( p \)th formulae, respectively, then the difference

\[
EST = ||\hat{y}_h(t_n) - y_h(t_n)||
\]

is an estimation of the local truncation error of the integration process. It is seen from lemma 3.2 that if the interpolation polynomial is of degree at least \( p \), then one can neglect the interpolation error and consider only the integration one, which can be estimated by the use of a Runge–Kutta formula pair. In our scheme a fifth degree Hermite interpolant is used together with the Runge–Kutta–Verner \((5,6)\) pair.

The strategy adopted for the stepsize control is to bound the local discretization error by a tolerance, \( \text{TOL} \), per unit step. If \( ||\cdot|| \) is the maximum norm, \( h_{\text{old}} \) is most recent stepsize already taken, \( h_{\text{new}} \) is the next step to be taken and \( h_{\text{min}} \) is the mini-
minimum step allowed by the routine. then the stepsize control algorithm is as follows:

After each integration step:

1. Compute the estimation \( EST \) per unit step, that is \( \| \hat{y}_h(t_n) - y(t_n) \| / h_{old} \).

2. If \( EST \leq TOL \), then
   
   \[
   \text{factor} = \min \{ 1.5, 0.9 \times (TOL/EST)^{1/6} \}.
   \]
   \[
   h_{new} = h_{old} \times \text{factor},
   \]
   \[
   n = n + 1,
   \]
   go to start.

3. If \( TOL < EST \), then
   
   If the number of successive failures is less than 3, then
   
   \[
   \text{factor} = \min \{ 1.5, 0.9 \times (TOL/EST)^{1/6} \},
   \]
   \[
   h_{new} = h_{old} \times \text{factor}
   \]
   go to start.

   If the number of successive failures is greater or equal to 3, then
   
   if \( h_{old} \times 0.5 > h_{min} \), then
   
   \[
   h_{new} = h_{old} \times 0.5,
   \]
   go to start
   
   if \( h_{old} \times 0.5 \leq h_{min} \) then
   
   \[
   h_{new} = h_{min}
   \]
   \[
   n = n + 1
   \]
   go to start
4.3 Numerical Results

In this section, four examples are used to illustrate the accuracy and the cost of our scheme. The numerical results of the test problems are given in tables, where the following abbreviations are used.

**TOL**: tolerance for the maximum norm of the error estimate.

**NFE**: number of function evaluations.

**TIME**: the total computing time in CPU seconds to solve the problem by the computer AMDAHL 4370 at the University of Ottawa.

**MRE**: the maximum relative error of the solution components at the final integration point \( t_f \).

**MAE**: the maximum absolute error of the solution components at the final integration point.

The exact solution is needed to compute the errors. If there is no exact solution, a reference solution obtained by numerical integration of the DDEs with very high precision is taken as an approximation to the exact one.

**Example 4.1 (Constant delay)**

The following well-known delay problem from biology

\[
\begin{align*}
y'(t) &= -3y(t-1)(1 + y(t)) \quad \text{for } t \geq 0, \\
y(t) &= t \quad \text{for } -1 \leq t \leq 0,
\end{align*}
\]  

(4.1)

has been taken from Bellen and Zennaro [1]. The numerical results of this problem at \( t_f = 20 \) for different values of the absolute error tolerance are given in table 4.1. The calculation of the error is obtained by comparison with the reference solution \( y(t_f) = 4.671437497500 \). Tolerances of \( 1E - 03 \) to \( 1E - 05 \) are not used since the
Table 4.1: Numerical results for example 4.1

<table>
<thead>
<tr>
<th>TOL</th>
<th>MAE</th>
<th>NFE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1E-06</td>
<td>5.40E-03</td>
<td>2318</td>
<td>0.35</td>
</tr>
<tr>
<td>1E-07</td>
<td>4.88E-04</td>
<td>3361</td>
<td>0.64</td>
</tr>
<tr>
<td>1E-08</td>
<td>2.36E-05</td>
<td>4782</td>
<td>1.16</td>
</tr>
<tr>
<td>1E-09</td>
<td>9.16E-06</td>
<td>7267</td>
<td>2.45</td>
</tr>
<tr>
<td>1E-10</td>
<td>3.59E-06</td>
<td>11068</td>
<td>5.31</td>
</tr>
<tr>
<td>1E-11</td>
<td>5.95E-07</td>
<td>17172</td>
<td>12.26</td>
</tr>
<tr>
<td>1E-12</td>
<td>6.94E-08</td>
<td>26874</td>
<td>25.72</td>
</tr>
</tbody>
</table>

routine failed to construct a 5th degree extrapolation polynomial to locate the first jump discontinuity.

Example 4.2 (State dependent delay)

In [22], the following DDE is used as a test problem

\[ y'(t) = \frac{1}{t} y(t) y(\log(y(t))) \quad \text{for} \quad t \geq 1, \]
\[ y(t) = 1 \quad \text{for} \quad 0 \leq t \leq 1. \quad (4.2) \]

The analytic solution of (4.2) is given by

\[ y(t) = \begin{cases} 
    t & \text{if } 1 \leq t \leq e \\
    \exp(t/e) & \text{if } e \leq t \leq e^2 \\
    \left(\frac{e}{3 - \log(t)}\right)^e & \text{if } e^2 \leq t \leq R_3 
\end{cases} \]

where \( R_3 = \exp[3 - \exp(1 - e)] \) is the third derivative jump discontinuity. Table 4.2 contains the numerical results for this state dependent delay problem. The error is computed on the interval \([1, e_3]\).

Example 4.3 (Variable time delay)

The following problem:

\[ y'(t) = \frac{t-1}{t} y(t) y(t - \log(t) - 1) \quad \text{for} \quad t \geq 1, \]
\[ y(t) = 1 \quad \text{for} \quad 0 \leq t \leq 1, \quad (4.3) \]
Table 4.2: Numerical results for example 4.2

<table>
<thead>
<tr>
<th>TOL</th>
<th>MRE</th>
<th>NFE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1E - 03</td>
<td>1.98E - 06</td>
<td>417</td>
<td>0.042</td>
</tr>
<tr>
<td>1E - 04</td>
<td>1.75E - 06</td>
<td>449</td>
<td>0.046</td>
</tr>
<tr>
<td>1E - 05</td>
<td>1.57E - 06</td>
<td>470</td>
<td>0.048</td>
</tr>
<tr>
<td>1E - 06</td>
<td>1.10E - 06</td>
<td>526</td>
<td>0.053</td>
</tr>
<tr>
<td>1E - 07</td>
<td>2.02E - 07</td>
<td>659</td>
<td>0.066</td>
</tr>
<tr>
<td>1E - 08</td>
<td>1.79E - 08</td>
<td>820</td>
<td>0.084</td>
</tr>
<tr>
<td>1E - 09</td>
<td>8.90E - 10</td>
<td>1156</td>
<td>0.123</td>
</tr>
<tr>
<td>1E - 10</td>
<td>1.67E - 10</td>
<td>1667</td>
<td>0.195</td>
</tr>
<tr>
<td>1E - 11</td>
<td>1.08E - 10</td>
<td>2451</td>
<td>0.304</td>
</tr>
<tr>
<td>1E - 12</td>
<td>6.17E - 13</td>
<td>3697</td>
<td>0.507</td>
</tr>
</tbody>
</table>

has been taken from Bellen-Zennaro [1]. As it was said earlier, the second and third derivative jump discontinuities of the solution occur at ξ₁ and ξ₂, respectively, where ξ₁ and ξ₂, are given in the example 4 of section 4.2. The explicit solution of (5.3) is:

\[
y(t) = \begin{cases} 
\exp\left(\frac{t-1}{t}\right) & \text{if } 1 \leq t \leq \xi_1 \\
\exp(1 + \int_{\xi_1}^{t} \frac{z-1}{z} \exp(x - \log(x - \log(x) - 1)) - 2 \, dx) & \text{if } \xi_1 \leq t \leq \xi_2
\end{cases}
\]

The numerical results corresponding to this problem are given in table 4.3. The relative error at \( t_f = \xi_2 \) is computed with respect to the reference solution \( y(t_f) = 76.3734725693768056269 \).

**Example 4.4 (Constant delay system)**

The following problem:

\[
y_1'(t) = y_2(t),
\]

\[
y_2'(t) = -y_2(t - 0.5)y_2(t)^2(t - 0.5), \quad 1 \leq t,
\]

with the initial conditions:

\[
y_1(t) = \log t,
\]
Table 4.3: Numerical results for example 4.3

<table>
<thead>
<tr>
<th>TOL</th>
<th>MRE</th>
<th>NFE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1E - 06$</td>
<td>$3.77E - 04$</td>
<td>316</td>
<td>0.03</td>
</tr>
<tr>
<td>$1E - 07$</td>
<td>$2.94E - 04$</td>
<td>344</td>
<td>0.03</td>
</tr>
<tr>
<td>$1E - 08$</td>
<td>$4.98E - 08$</td>
<td>575</td>
<td>0.05</td>
</tr>
<tr>
<td>$1E - 09$</td>
<td>$6.38E - 09$</td>
<td>792</td>
<td>0.07</td>
</tr>
<tr>
<td>$1E - 10$</td>
<td>$1.77E - 10$</td>
<td>1303</td>
<td>0.13</td>
</tr>
<tr>
<td>$1E - 11$</td>
<td>$2.20E - 11$</td>
<td>1947</td>
<td>0.20</td>
</tr>
<tr>
<td>$1E - 12$</td>
<td>$1.25E - 12$</td>
<td>3025</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Table 4.4: Numerical results for example 4.4 at $t_f = 10$

<table>
<thead>
<tr>
<th>TOL</th>
<th>MAE</th>
<th>NFE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1E - 04$</td>
<td>$7.76E - 06$</td>
<td>442</td>
<td>0.09</td>
</tr>
<tr>
<td>$1E - 05$</td>
<td>$7.51E - 07$</td>
<td>491</td>
<td>0.10</td>
</tr>
<tr>
<td>$1E - 06$</td>
<td>$2.66E - 08$</td>
<td>575</td>
<td>0.12</td>
</tr>
<tr>
<td>$1E - 07$</td>
<td>$4.59E - 09$</td>
<td>659</td>
<td>0.14</td>
</tr>
<tr>
<td>$1E - 08$</td>
<td>$7.40E - 10$</td>
<td>771</td>
<td>0.16</td>
</tr>
<tr>
<td>$1E - 09$</td>
<td>$6.02E - 11$</td>
<td>953</td>
<td>0.20</td>
</tr>
<tr>
<td>$1E - 10$</td>
<td>$3.61E - 12$</td>
<td>1191</td>
<td>0.26</td>
</tr>
<tr>
<td>$1E - 11$</td>
<td>$2.53E - 13$</td>
<td>1618</td>
<td>0.38</td>
</tr>
<tr>
<td>$1E - 12$</td>
<td>$3.26E - 14$</td>
<td>2269</td>
<td>0.58</td>
</tr>
</tbody>
</table>

$$y_2(t) = \frac{1}{t}, \quad 0 \leq t \leq 1.$$  

has the smooth solution:

$$y_1(t) = \log t,$$

$$y_2(t) = \frac{1}{t}, \quad 1 \leq t.$$  

Two different values of $t_f$, 10 and 1000, are used for this problem, and the numerical results are shown in tables 4.4 and 4.5, respectively.
Table 4.5: Numerical results for example 4.4 at $t_f = 1000$

<table>
<thead>
<tr>
<th>TOL</th>
<th>MAE</th>
<th>NFE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1E - 07</td>
<td>4.68E - 09</td>
<td>21484</td>
<td>29.21</td>
</tr>
<tr>
<td>1E - 08</td>
<td>7.71E - 10</td>
<td>31088</td>
<td>60.37</td>
</tr>
<tr>
<td>1E - 09</td>
<td>6.18E - 11</td>
<td>45256</td>
<td>129.60</td>
</tr>
<tr>
<td>1E - 10</td>
<td>4.46E - 12</td>
<td>65948</td>
<td>279.23</td>
</tr>
<tr>
<td>1E - 11</td>
<td>1.49E - 12</td>
<td>96475</td>
<td>597.22</td>
</tr>
<tr>
<td>1E - 12</td>
<td>2.06E - 13</td>
<td>142324</td>
<td>1279.83</td>
</tr>
</tbody>
</table>

4.4 Comparison With Other Methods

In this section, the proposed scheme is compared with three other numerical methods for delay differential equations. The comparison is done with respect to precision and cost (i.e. the total number of function evaluations needed to solve the DDEs). These methods are:

1. **Bellen–Zennaro method** [1]. This method is based on a predictor-corrector mode that uses the one-step collocation method at $n$ Gaussian points and it provides a continuous extension of the numerical solution.

2. **Thompson software** [22]. This software uses continuously imbedded Runge-Kutta methods of Sarafyan. $C^1$ piecewise polynomial interpolants are used for the interpolation and the automatic localization of the derivative jump discontinuities of the solution.

3. **Oberle–Pesch methods** [8]. These methods, denoted by RKFR4 and RKFR7, are based on Runge-Kutta-Fehlberg formulae of the orders 4 and 7, respectively, together with Hermite interpolations of the appropriate degrees.
The examples of the previous section are used as comparison tests.

5.1 Comparison for Example 4.1 The results of the proposed scheme are listed in table 4.6. The precision of the numerical results depends upon the method. For a fixed tolerance, the proposed scheme is slightly more accurate and except for RKFR7, it is also less expensive than the other methods.

5.2 Comparison for Example 4.2
The results of the proposed scheme and the Skip Thompson software for example 4.2 are given in table 4.7. For this example, the amount of computation used by each method is approximately the same. However, the proposed scheme is much more accurate than the Skip Thompson software.

5.3 Comparison for Example 4.3
Table 4.8 contains the numerical results for example 4.3 obtained by Bellen–Zennaro and the proposed scheme, respectively. For this problem, it is clear that the Bellen–
Table 4.8: Comparison for example 4.3

<table>
<thead>
<tr>
<th>TOL</th>
<th>Bellen-Zennaro</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRE</td>
<td>NFE</td>
</tr>
<tr>
<td>1E-06</td>
<td>2.4E-07</td>
<td>821</td>
</tr>
<tr>
<td>1E-07</td>
<td>2.7E-08</td>
<td>1111</td>
</tr>
<tr>
<td>1E-08</td>
<td>5.6E-09</td>
<td>1611</td>
</tr>
<tr>
<td>1E-09</td>
<td>7.0E-10</td>
<td>2471</td>
</tr>
<tr>
<td>1E-10</td>
<td>7.7E-11</td>
<td>4016</td>
</tr>
</tbody>
</table>

Zennaro method is much more accurate than the proposed method, but the former is more expensive.
Chapter 5

Conclusion

The way how the derivative jump discontinuities are located is critical for the performance of any numerical DDEs solver. It is clear from the numerical results presented in the previous chapter that generally an adapted Runge–Kutta scheme for solving DDEs achieves the required accuracy. It is shown that the effect of approximating the delayed term is negligible and in general it is sufficient to use an interpolation polynomial of degree one less than the order of the method. Also the use of a Runge–Kutta formula pair in the numerical scheme provides an elegant way to estimate the local truncation error of the method and consequently to perform the stepsize control.

The actual algorithm is not the most efficient one; but it can be used as an example of how algorithms from ODEs can be modified to solve DDEs.

Finally, the numerical solution of DDEs should benefit from the existing softwares for ODEs since the complication arising from discontinuities and interpolation can be overcome without resorting to completely new algorithms.
Bibliography


Appendix A

Routine for Solving DDEs
PROGRAM SYSDEL
IMPLICIT DOUBLE PRECISION(A-H,L-Z)
INTEGER U,X,XX,KK,XM,MNR
DIMENSION T(30000),Y(2,30000),R(20),C(12),L(2,8),SS(2),S1(2),
+S2(2),S3(2),S4(2),S5(2),S6(2),S7(2),B(2),LL(2),A(20,20),F(2,30000)
TOL=1E-12
TEND= 1000.00D0
T(1)= 1.00D0
Y(1,1)= 0.00D0
Y(2,1)= 1.00D0
X=2
XM=30000
E=TOL
IND=1
R(1)=1.00D0
MNR=21.00D0
C(1) = 2.00D0
C(2) = (TOL)**(1.00D/6.D0)
C(5) = (TOL)**(1.00D/6.D0)
C(3) = (TOL)**(1.00D/5.D0)**0.1D0
C(4) = DMIN1(0.5D0,5.0D0*(TOL)**(1.00D/6.D0))
CALL DDES(X,XM,TOL,T(1),Y,TEND,CO,IND,R(1),MNR,L,B,S1,S2,S3,
+S4,S5,S6,S7,SS,R,A,LL,F,T)
STOP
END

FUNCTION AL(I,T,Y)
IMPLICIT DOUBLE PRECISION (A-H,L-Z)
DIMENSION Y(2),ALPHA(2)
GO TO (1,2) I
1 AL = 0.1D0
RETURN
2 AL = T-0.9D0
RETURN
END

FUNCTION PHI(I,T)
IMPLICIT DOUBLE PRECISION (A-H,L-Z)
GO TO (1,2) I
1 PHI = DLOG(T)
RETURN
2 PHI = 1.0D0/T
RETURN
C
C
FUNCTION G(I,T,Y,V)
IMPLICIT DOUBLE PRECISION (A-H,L-Z)
DIMENSION Y(2),V(2)
GO TO (1,2) I
1
C = Y(2)
RETURN
2
G = -V(2) * Y(2)**2 * (I - 0.9D0)
RETURN
END
C
C
SUBROUTINE DDES(X,XM,TOL,DD,Y,TEND,IND,RR,MNR,L,B,S1,
+ S2,S3,S4,S5,S6,S7,SS,R,LL,F,T)
IMPLICIT DOUBLE PRECISION(A-H,L-Z)
INTEGER U,X,XX,IX,XM,MNR
DIMENSION Y(XM),T(XM),F(X,XM),C(12),L(X,7)
+ d(X),S5(X),S1(X),S2(X),S3(X),
+ S4(X),S5(X),S6(X),S7(X),R(MNR),A(MNR,MNR),LL(2)
EXTERNAL G
EXTERNAL AL
EXTERNAL PHI
C***************************************************************************
C PURPOSE:
C THIS CODE IS BASED ON RUNGE-KUTTA-VERNER'S(5,6) FORMULA PAIR
C (RKV(5,6) PAIR)
C AND A 3-POINT HERMITE INTERPOLATION POLYNOMIAL. IT APPROXIMATES THE
C SOLUTION OF A SYSTEM OF A FIRST ORDER NON-VANISHING LAG STATE DEPEND-
C ENT DELAY DIFFERENTIAL EQUATIONS WITH INITIAL CONDITIONS.
C MANY OPTIONS ARE AVAILABLE TO THE USER, INCLUDING TWO KINDS
C OF ERROR CONTROL (ABSOLUTE AND RELATIVE ERROR CONTROL) AND TWO OPT-
C IONS CONCERNING WHETHER OR NOT THE DERIVATIVE JUMP DISCONTINUITIES OF
C THE SOLUTION SHOULD BE LOCATED. HOWEVER THE USER IS ADVISED TO USE
C THE JUMP DISCONTINUITIES OPTION IN ORDER TO IMPROVE THE ACCURACY OF
C THE NUMERICAL APPROXIMATIONS. THE EVALUATION OF THE DELAYED TERM
C AL(T,Y(T)) BY A 3-POINT HERMITE INTERPOLATION POLYNOMIAL Requires
C THE STORAGE OF THE TIME AND THE SOLUTION AS WELL AS ITS FIRST
C DERIVATIVE DURING THE INTEGRATION PROCESS.
C THIS CODE SOLVES A SYSTEM OF X EQUATIONS, EACH EQUATION WITH
C ONE DELAY FUNCTION. NOTE THAT MULTIPLE DELAY SYSTEM CANNOT
C BE HANDLED DIRECTLY BY THIS CODE.
C
C***************************************************************************
THE USER MUST SPECIFY EACH OF THE FOLLOWING ENTRIES

** X : THE NUMBER OF EQUATIONS IN THE SYSTEM.
** T(1): THE INITIAL TIME OF THE INTEGRATION.
** Y : THE INITIAL VALUES OF THE SOLUTIONS, I.E. Y(1,1),...,Y(X,1).
** TEND: THE FINAL INTEGRATION TIME, I.E. THE TIME TO WHICH THE INTEGRATION IS TO BE CARRIED OUT.
** TOL: TOLERANCE, THE CODE USES A STANDARD STEP-SIZE CONTROL AND ATTEMPTS TO KEEP THE GLOBAL ERROR PROPORTIONAL TO THE TOLERANCE.
** IND : INDICATOR FOR THE ROOTFINDING OPTION.
** IND = 1: THE CODE USES THE ROOTFINDING OPTION TO LOCATE THE DERIVATIVE JUMP DISCONTINUITIES.
** IND = 2: THE ROOTFINDING OPTION IS NO LONGER USED BY THE CODE AND CONSEQUENTLY THE DISCONTINUITIES CAN NOT BE LOCATED. THIS MAY LEAD TO A LACK OF ACCURACY IN THE NUMERICAL SOLUTION.
** MNR : THE MAXIMUM NUMBER OF POSSIBLE DERIVATIVE JUMP DISCONTINUITIES TO BE LOCATED BY THE CODE DURING THE INTEGRATION PROCESS.
** C : THE COMMUNICATION VECTOR.
** C(1) = ERROR CONTROL INDICATOR.
** C(1)=1 THE CODE USES THE ABSOLUTE ERROR CONTROL.
** C(1)=2 THE CODE USES THE RELATIVE ERROR CONTROL.
** C(2) = THE STARTING STEPSIZE TO BE SUPPLIED BY THE USER. IF IT IS NECESSARY, THE PROGRAM RETURNS WITH A MESSAGE TO CHANGE THE VALUE OF C(2).
** C(5) = THE ORIGINAL STARTING STEP-SIZE; IT IS FIXED DURING THE INTEGRATION TIME AND MUST NOT BE CHANGED.
** C(3) = THE MINIMUM STEP-SIZE THAT IS ALLOWED TO BE TAKEN BY THE CODE.
** C(4) = THE MAXIMUM STEP-SIZE THAT IS ALLOWED TO BE TAKEN BY THE CODE.
** THE OTHER COMPONENTS OF C ARE AUTOMATICALLY USED BY THE CODE.
** G : FUNCTION G(I,T,Y,V) WHERE:
** G IS THE X-VALUED VECTOR FUNCTION.
** Y IS THE SOLUTION OF THE SYSTEM.
** V IS THE SOLUTION AT THE DELAYED TERM.
** AL : ALPHA(I,T,Y) IS THE X-VALUED DELAY VECTOR FUNCTION.
** PHI : PHI(I,T) IS THE X-VALUED INITIAL VECTOR FUNCTION.
**
C----------------------------------------------------------------------
LL(1)=TOL
U=1
XX=1
I=1
C(7) = C(2)
C(8) = 0.0D0
C(9) = 1.0D0
C(11)=1.0D0
C(12)=1.0D0
T(1)=DD
R(1)=RR
E=TOL
CALL CPTIME(0,100)
CALL CPTIME(1,100)
DO 1965 K=1,X
F(K,I)= 0.0D0
1965 CONTINUE
DO 912 K=1,7
  DO 913 J=1,X
    A(J,K)=0.0D0
  913 CONTINUE
912 T(K)=T(1)
911 CONTINUE
C
C INITIALIZE, FIND THE SLu Property OF THE SOLUTION
C
1 DO 71 K=1,X
B(K)=Y(K,I)
71 CONTINUE
DO 7 K=1,X
D=T(1)
CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,F(K,I),J)
7 CONTINUE
C
C CALCULATE HMAG BY THE USE OF THE STANDRAD FORMULA OF THE STEP-SIZE
C CONTROL
C
IF(E.EQ.0.0D0) E=TOL
IF (E.LT.TOL) THEN
14 CONTINUE
FACTOR=DMIN1(1.5D0,0.9D0*(TOL/E)**(1.0D0/6.0D0))
C(7)=C(7)=FACTOR
C
C CHECK AGAINST HMAX, HMIN

61
ENDIF
C(7)=D MAX1(C(3),C(7))
C(7)=D MIN1(C(4),C(7))

C COMPUTE THE REMAINING 7 STAGES OF THE ADAPTED RKV(5,5) PAIR.

DO 8 K=1,X
B(K)=Y(K,I)+C(7)*F(K,I)*(1.D0/6.D0)
8 CONTINUE
D=T(I)+C(7)*(1.D0/6.D0)
DO 9 K=1,X
CALL EVAL(I,K,X,XM,DB,LY,F,V,L(K,1),J)
9 CONTINUE
DO 13 K=1,X
B(K)=Y(K,I)+C(7)*((F(K,I)*4.D0/75.D0)+16.D0/75.D0*L(K,1))
13 CONTINUE
D=T(I)+C(7)*(4.D0/15.D0)
DO 15 K=1,X
CALL EVAL(I,K,X,XM,DB,LY,F,V,L(K,2),J)
15 CONTINUE
DO 35 K=1,X
B(K)=Y(K,I)+C(7)*((F(K,I)*5.D0/6.D0)-L(K,1)*8.D0/3.D0+L(K,2)
  +*5.0D0/2.D0)
35 CONTINUE
D=T(I)+(2.D0/3.D0)*C(7)
DO 16 K=1,X
CALL EVAL(I,K,X,XM,DB,LY,F,V,L(K,3),J)
16 CONTINUE
DO 17 K=1,X
B(K)=Y(K,I)+C(7)*(-F(K,I)*165.D0/64.D0)+L(K,1)*55.D0/6.D0
  +L(K,2)*425.D0/64.D0+L(K,3)*85.D0/96.D0)
17 CONTINUE
DO 18 K=1,X
D=T(I)+C(7)*5.D0/6.D0
CALL EVAL(I,K,X,XM,DB,LY,F,V,L(K,4),J)
18 CONTINUE
DO 19 K=1,X
D=T(I)+C(7)
B(K)=Y(K,I)+C(7)*((F(K,I)*12.D0/5.D0)-8.0D0*L(K,1)+4015.D0/612.D0=
  L(K,2)-11.0D0/36.D0*L(K,3)+88.0D0/255.D0*L(K,4))
+L(K,2)-11.0D0/36.D0*L(K,3)+88.0D0/255.D0*L(K,4))
19 CONTINUE
DO 21 K=1,X
CALL EVAL(I,K,X,XM,DB,LY,F,V,L(K,5),J)
21 CONTINUE
DO 22 K=1,X
B(K)=Y(K,1)+C(7)*(-F(K,1)*8263.D0/15000.D0+124.D0/75.D0*L(K,1)
+L(K,2)*643.D0/680.D0-81.D0/250.D0*L(K,3)+2484.D0/10625.D0=
+L(K,4))
22 CONTINUE
D=T(I)+C(7)*(1.D0/15.D0)
DO 23 K=1,X
CALL EVAL(I,K,XM,D,B,T,Y,F,V,L(K,6),J)
23 CONTINUE
DO 24 K=1,X
B(K)=Y(K,1)+C(7)*(F(K,1)*3501.D0/1720.D0-L(K,1)*300.D0/43.D0*L
+L(K,2)*297275/52632.D0-L(K,3)*319.D0/2322.D0+L(K,4)*24068.D0/
+84065.D0-L(K,6)*3850.D0/26703.D0)
24 CONTINUE
D=T(I)+C(7)
DO 25 K=1,X
CALL EVAL(I,K,XM,D,B,T,Y,F,V,L(K,7),J)
25 CONTINUE
C
C COMPUTE THE SOLUTION COMPONENTS.
C
DO 26 K=1,X
Y(K,1)=Y(K,1)+C(7)*(F(K,1)*3.D0/40.D0*L(K,2)*875.D0/2244.D0+=
+23.D0/72.D0*L(K,3)+
+264.D0/1955.D0*L(K,4)+1!GDO/11592.D0*L(K,6)+43.D0/616.D0*L(K,7))
26 CONTINUE
C
C COMPUTE THE ERROR ESTIMATE BY USING THE RKV(5,6) PAIR.
C
DO 27 K=1,X
L(K,8)=F(K,1)*(3.D0/40.D0-13.D0/160.D0)+(875.D0/2244.D0-2375.D0 /
+5984.D0)*L(K,2)+(23.D0/72.D0-5.D0/16.D0)*L(K,3)+
+(264.D0/1955.D0-12.D0/85)*L(K,4)
+3.D0/44.D0*L(K,5)+125.D0/11592.D0*L(K,6)+43.D0/616.D0*L(K,7))
C
C COMPUTE THE ERROR ESTIMATE BY USING ABSOLUTE ERROR CONTROL.
C
27 CONTINUE
E=DABS(L(1,8))
IF(C(1).EQ.1.D0) THEN
DO 28 K=1,X
E=DMAX1(E,DABS(L(K,8)))
28 CONTINUE
ENDIF
C
C COMPUTE THE ERROR ESTIMATE BY USING THE RELATIVE ERROR CONTROL.
C
IF(C(1).EQ.2.0D0) THEN
  IF(Y(1,I).NE.0.0D0) THEN
    E=DAABS(L(1,8)/Y(1,I))
  ENDIF
  E=DAABS(L(1,8))
  DO 29 K=1,X
    IF(Y(K,I).NE.0.0D0) THEN
      E=DMAX1(E,DAABS(L(K,8)/Y(K,I)))
    ENDIF
  29 CONTINUE
ENDIF
E=DAABS(E)
LL(2)=E+1.0D-16
  IF(LL(2)/LL(1).GT.100) THEN
    IF(C(11).EQ.1.0D0) THEN
      C(7)=0.0075D0
      C(11)=C(11)+1.0D0
      GO TO 1
    ENDIF
  ENDIF
C(11)=1.0D0
T(I+1)=T(I)+C(7)

C IN CASE WHERE THE ROOTFINDING OPTION IS USED AND THE NUMBER OF DISCONT
C INUITIES ALREADY LOCATED IS LESS THAN MNR, CHECK IF THERE IS A
C CHANGE IN SIGN OF THE SWITCHING FUNCTION.
C
IF(XX.LT.MNR) THEN
  IF (IND.EQ.1) THEN
    DO 499 K=1,X
      SS(K)=Y(K,I+1)
      S1(K)=Y(K,I)
    499 CONTINUE
    KK=1
    U=1
  31 IF((AL(KK,T(I),S1)-R(U))*(AL(KK,T(I+1),SS)-R(U)).GE.0.0D0) THEN
      IF(U.LE.XX-1) THEN
        U=U+1
        GO TO 31
      ENDIF
      IF(KK.LE.X-1) THEN
        KK=KK+1
  31 CONTINUE
GO TO 31
  ENDIF

C
C SINCE THERE IS NO DISCONTINUITY IN THE CURRENT STEP, GO TO 12 TO
C DECIDE WHETHER THE STEP IS TO BE REJECTED OR ACCEPTED.
C
GO TO 12
  ENDIF

C
C A DISCONTINUITY HAS OCCURRED IN THE CURRENT STEP; COMPUTE THE
C APPROPRIATE NUMBER OF THE SWITCHING FUNCTION AT THE PREVIOUS TIME TO BE
C USED BY THE ROOTFINDING PROCESS.
C
IF(I.LT.4) THEN
  WRITE(*,1992)
  1992 FORMAT(/,1X,61('**'),/,'2(1X,'**',59X,'**',/),1X,'**',5X,
          '** NUM. OF STEPS IS LESS THAN 4, PLEASE REDUCE C(2)**,5X,**,
          '/,'2(1X,'**',59X,'**',/),1X,61('**'),/)
  STOP
ENDIF

A(U,1)=AL(KK,T(I),S1)-R(U)
DO 501 K=1,X
S2(K)=Y(K,I-1)
501 CONTINUE
A(U,2)=AL(KK,T(I-1),S2)-R(U)
DO 502 K=1,X
S3(K)=Y(K,I-2)
502 CONTINUE
A(U,3)=AL(KK,T(I-2),S3)-R(U)
DO 503 K=1,X
S4(K)=Y(K,I-3)
503 CONTINUE
A(U,4)=AL(KK,T(I-3),S4)-R(U)
T1=T(I)
T2=T(I-1)
T3=T(I-2)
T4=T(I-3)
T5=0.0DO
T6=0.0DO
T7=0.0DO
IF(I.EQ.4) GO TO 11
DO 504 K=1,X
S5(K)=Y(K,I-4)
504 CONTINUE
A(U,5)=AL(KK,T(I-4),S5)-R(U)
T5=T(I-4)
IF(I.EQ.5) GO TO 11
DO 505 K=1,X
S6(K)=Y(K,I-5)
505 CONTINUE
A(U,6)=AL(KK,T(I-5),S6)-R(U)
T6=T(I-5)
IF(I.EQ.6) GO TO 11
DO 506 K=1,X
S7(K)=Y(K,I-6)
506 CONTINUE
A(U,7)=AL(KK,T(I-6),S7)-R(U)
T7=T(I-6)
C
C GO TO 11 TO LOCATE THE DERIVATIVE JUMP DISCONTINUITY.
C
GO TO 11
ENDIF
ENDIF
12 CONTINUE
C
C IF THE ESTIMATE IS GREATER THAN THE TOLERANCE AND THE NUMBER OF
C SUCCESSIVE FAILURES IS LESS THAN OR EQUAL TO 2, THEN REDUCE THE
C STEP SIZE BY A HALF AND RECOMPUTE THE NUMERICAL APPROXIMATION.
C
IF(E.GT.TOL) THEN
IF(C8.LE.2.0D0) THEN
FACTOR=DMIN1(1.5d0,0.9d0*(tol/e)**(1.0d0/6.0d0))
C7=FACTOR*C7
C8=C8+1.0D0
LL(1)=LL(2)
GO TO 1
ENDIF
C
C IF THE NUMBER OF SUCCESSIVE FAILURES EXCEEDS 2, THEN KEEP HALF-
CING THE STEP SIZE UNTIL A SUCCESSFUL STEP IS OBTAINED OR THE
C MINIMUM STEP IS REACHED.
C
IF(C8.GT.2.0D0) THEN
IF(C7.GT.0.5D0.GT.C3)
C7=C7*0.5
GO TO 1
ENDIF
T(I+1)=T(I)+C7
ENDIF
ENDIF
C
C IN CASE WHERE THE STARTING STEP SIZE WAS REDUCED, RETURN TO THE
C ORIGINAL C(2)
C
IF(I.GT.4) THEN
  IF(C(2).LT.C(5)) THEN
    C(2)=C(5)
    C(7)=C(2)
  ENDFI
  IF(T(I+1).LT.TEND) THEN
    I=I+1
    LL(1)=LL(2)
    GO TO 1
  ENDFI
  I=I+1
  LL(1)=LL(2)
  GO TO 20
11 CONTINUE
  CALL NEWDIF(I,A(U,1),A(U,2),A(U,3),A(U,4),A(U,5),A(U,6)
  +,A(U,7),T1,T2,T3,T4,T5,T6,T7,D1,D2,D3,D4,D5,D6)
C
  CALL BISECT(D1,D2,D3,D4,D5,D6,A(U,1),T1,T2,T3,T4,T5,T6,R1)
  C(7)=R1-T(I)
C
C STORE THE POINT OF DISCONTINUITY AND ADD 1 TO THE NUMBER OF ROOTS.
C
  R(XX+1)=R1
  XX=XX+1
C
C GO TO 10 TO COMPUTE THE SOLUTION AT THE JUMP POINT DISCONTINUITY.
C
  GO TO 10
10 DO 72 K=1,X
  B(K)=Y(K,I)
  72 CONTINUE
  DO 37 K=1,X
    D=T(I)
    CALL EVAL(I,K,X,XM,T(I),B,T,Y,F,V,F(K,I),J)
    37 CONTINUE
    DO 38 K=1,X
      B(K)=Y(K,I)+C(7)*F(K,I)*(1.0D0/6.0D0)
      38 CONTINUE
      D=T(I)+C(7)*(1.0D0/6.0D0)
DO 39 K=1,X
CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,1),J)
39 CONTINUE
DO 89 K=1,X
B(K)=Y(K,I)+C(7)*(F(K,I)*4.0D0/75.0D0+16.0D0/75.0D0*L(K,1))
89 CONTINUE
D=T(I)+C(7)*(4.0D0/15.0D0)
DO 40 K=1,X
CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,2),J)
40 CONTINUE
DO 41 K=1,X
B(K)=Y(K,I)+C(7)*(F(K,I)*5.0D0/6.0D0-L(K,1)*8.0D0/3.0D0+L(K,2)
  +5.0D0/2.0D0)
41 CONTINUE
D=T(I)+(2.0D0/3.0D0)*C(7)
DO 42 K=1,X
CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,3),J)
42 CONTINUE
DO 43 K=1,X
B(K)=Y(K,I)+C(7)*(-F(K,I)*165.0D0/64.0D0+L(K,1)*55.0D0/6.0D0
  -L(K,2)*425.0D0/64.0D0+L(K,3)*85.0D0/96.0D0)
43 CONTINUE
DO 44 K=1,X
D=T(I)+C(7)*5.0D0/6.0D0
CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,4),J)
44 CONTINUE
DO 45 K=1,X
D=T(I)+C(7)
B(K)=Y(K,I)+C(7)*(F(K,I)*12.0D0/5.0D0-8.0D0*L(K,1)+4015.0D0/612.0D0*
  L(K,2)-11.0D0/36.0D0*L(K,3)+88.0D0/255.0D0*L(K,4))
45 CONTINUE
DO 46 K=1,X
CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,5),J)
46 CONTINUE
DO 47 K=1,X
B(K)=Y(K,I)+C(7)*(-F(K,I)*8263.0D0/15000.0D0+124.0D0/75.0D0*L(K,1)
  -L(K,2)*643.0D0/680.0D0-81.0D0/250.0D0*L(K,3)+2484.0D0/10625.0D0*
  L(K,4))
47 CONTINUE
D=T(I)+C(7)*(1.0D0/15.0D0)
DO 48 K=1,X
CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,6),J)
48 CONTINUE
DO 49 K=1,X
B(K)=Y(K,I)+C(7)*(F(K,I)*3501.0D0/1720.0D0-L(K,1)*300.0D0/43.0D0+L
\[
(K,2) = 297275/52632.0 - L(K,3) * 319.0 / 2322.0 = L(K,4) * 24068.0 / \\
84065.0 = L(K,6) * 3850.0 / 26703.0
\]

49

\begin{verbatim}
CONTINUE
D = T(I) + C(7)
DO 50 K = 1, X
    CALL EVAL(I, K, X, XM, D, B, T, Y, F, V, L(K,7), J)
50 CONTINUE
\end{verbatim}

C COMPUTE THE SOLUTION COMPONENTS.

\begin{verbatim}
DO 51 K = 1, X
    Y(K, I+1) = Y(K, I) + C(7) * \left( F(K, I) * 3.0 / 40.0 + L(K, 2) * 875.0 / 2244.0 + \\
        23.0 / 72.0 * L(K, 3) + \\
        264.0 / 1955.0 * L(K, 4) + 125.0 / 11592.0 * L(K, 6) + 43.0 / 616.0 * L(K, 7) \right)
51 CONTINUE
\end{verbatim}

C COMPUTE THE ERROR ESTIMATE BY USING THE RKV(5,6) PAIR.

\begin{verbatim}
DO 52 K = 1, X
    L(K, 8) = \left( F(K, I) * (3.0 / 40.0 - 13.0 / 160.0) * (875.0 / 2244.0 - 2375.0 / 5984.0) * L(K, 2) + \\
        (23.0 / 72.0 - 5.0 / 16.0) * L(K, 3) + \\
        (264.0 / 1955.0 - 12.0 / 85) * L(K, 4) + \\
        -3.0 / 44.0 * L(K, 5) + 125.0 / 11592.0 * L(K, 6) + 43.0 / 616.0 * L(K, 7) \right)
\end{verbatim}

C COMPUTE THE ERROR ESTIMATE BY USING THE ABSOLUTE ERROR CONTROL.

\begin{verbatim}
52 CONTINUE
E = DABS(L(1, 8))
IF(C(1).EQ.1.0D0) THEN
    DO 53 K = 1, X
        E = DMAX1(E, DABS(L(K, 8)))
53 CONTINUE
ENDIF
\end{verbatim}

C COMPUTE THE ERROR ESTIMATE BY USING THE RELATIVE ERROR CONTROL.

\begin{verbatim}
IF(C(1).EQ.2.0D0) THEN
    IF(Y(K, I).NE.0.0D0) THEN
        E = DABS(L(1, 8)) / Y(K, I)
    ENDIF
    E = DMAX1(E, DABS(I(K, 8)) / Y(K, I))
\end{verbatim}

69
ENDIF
54 CONTINUE
ENDIF
E=DABS(E)
T(I+1)=T(I)+C(7)
LL(1)=E+1.0D-16
I=I+1
GO TO 20
20 IF(T(I).GE.TEND) THEN
C(7)=TEND-T(I-1)
T(I)=T(I-1)
DO 100 K=1,X
   Y(K,I)=Y(K,I-1)
100 CONTINUE
ENDIF
DO 73 K=1,X
   B(K)=Y(K,I)
73 CONTINUE
DO 57 K=1,X
   D=T(I)
   CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,F(K,I),J)
57 CONTINUE
DO 58 K=1,X
   B(K)=Y(K,I)+C(7)*F(K,I)*(1.0D0/6.0D0)
58 CONTINUE
D=T(I)+C(7)*(1.0D0/6.0D0)
DO 59 K=1,X
   CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,1),J)
59 CONTINUE
DO 519 K=1,X
   B(K)=Y(K,I)+C(7)*(F(K,I)*4.0D0/75.0D0+16.0D0/75.0D0*L(K,1))
519 CONTINUE
D=T(I)+C(7)*(4.0D0/15.0D0)
DO 60 K=1,X
   CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,2),J)
60 CONTINUE
DO 61 K=1,X
   B(K)=Y(K,I)+C(7)*(F(K,I)*5.0D0/6.0D0-L(K,1)*8.0D0/3.0D0+L(K,2)
   **5.0D0/2.0D0)
61 CONTINUE
D=T(I)+(2.0D0/3.0D0)*C(7)
DO 62 K=1,X
   CALL EVAL(I,K,X,XM,D,B,T,Y,F,V,L(K,3),J)
62 CONTINUE
DO 63 K=1,X
\[ B(K) = Y(K,I) + C(7) \times (-F(K,I) \times 165.0D0/64.0D0 + L(K,1) = 55.0D0/6.0D0) \]

\[ + -L(K,2) \times 425.0D0/64.0D0 + L(K,3) = 85.0D0/96.0D0) \]

63
CONTINUE
DO 64 K=1,X
D=T(I)+C(7)*5.0D0/6.0D0
CALL EVAL(I,K,XM,D,B,T,Y,F,V,L(K,4),J)
64
CONTINUE
DO 65 K=1,X
D=T(I)+C(7)
B(K) = Y(K,I) + C(7) \times (F(K,I) \times 12.0D0/5.0D0 - 8.0D0 \times L(K,1) + 4015.0D0/612.0D0 =

\[ + -L(K,2) \times 11.0D0/36.0D0 \times L(K,3) + 88.0D0/255.0D0 \times L(K,4) \]

65
CONTINUE
DO 66 K=1,X
CALL EVAL(I,K,XM,D,B,T,Y,F,V,L(K,5),J)
66
CONTINUE
DO 67 K=1,X
B(K) = Y(K,I) + C(7) \times (-F(K,I) \times 8263.0D0/15000.0D0 + 124.0D0/75.0D0 \times L(K,1) =

\[ + -L(K,2) \times 643.0D0/680.0D0 - 81.0D0/250.0D0 \times L(K,3) + 2484.0D0/10625.0D0 =

\[ + -L(K,4) \]

67
CONTINUE
D=T(I)+C(7)*(1.0D0/15.0D0)
DO 68 K=1,X
CALL EVAL(I,K,XM,D,B,T,Y,F,V,L(K,6),J)
68
CONTINUE
DO 69 K=1,X
B(K) = Y(K,I) + C(7) \times (F(K,I) \times 3501.0D0/1720.0D0 - L(K,1) = 300.0D0/43.0D0 + L

\[ + -297275/52632.0D0 - L(K,3) = 319.0D0/2322.0D0 + L(K,4) = 24068.0D0/84065.0D0 =

\[ + -L(K,6) \times 3850.0D0/26703.0D0 \]

69
CONTINUE
D=T(I)+C(7)
DO 80 K=1,X
CALL EVAL(I,K,XM,D,B,T,Y,F,V,L(K,7),J)
80
CONTINUE

C
COMPUTE THE SOLUTION COMPONENTS.
C
DO 81 K=1,X
Y(K,I+1) = Y(K,I) + C(7) \times (F(K,I) \times 3.0D0/40.0D0 + L(K,2) = 875.0D0/2244.0D0 +

\[ + -23.0D0/72.0D0 \times L(K,3) =

\[ + -264.0D0/1955.0D0 \times L(K,4) + 125.0D0/11592.0D0 \times L(K,6) + 43.0D0/615.0D0 \times L(K,7) \]

81
CONTINUE
C(9) = C(9) + 7.0D0
C
COMPUTE THE ERROR ESTIMATE BY USING THE RKV(5,6) PAIR.
C

DO 82 K=1,X
L(K,8)=(F(K,I)*(3.0D0/40.0D0-13.0D0/160.0D0)+(875.0D0/2244.0D0-2375.0D0
+/5984.0D0)*L(K,2)+(23.0D0/72.0D0-5.0D0/16.0D0)*L(K,3)+
+(264.0D0/1955.0D0-12.0D0/85)*L(K,4)
+-3.0D0/44.0D0*L(K,5)+125.0D0/11592.0D0*L(K,6)+43.0D0/616.0D0*L(K,7))
C
C COMPUTE THE ERROR ESTIMATE BY USING THE ABSOLUTE ERROR CONTROL.
C
82 CONTINUE
E=DABS(L(1,8))
IF(C(1).EQ.1.0D0) THEN
DO 83 K=1,X
E=DMAX1(E,DABS(L(K,8)))
83 CONTINUE
ENDIF
C
C COMPUTE THE ERROR ESTIMATE BY USING THE RELATIVE ERROR CONTROL.
C
IF(C(1).EQ.2.0D0) THEN
IF(Y(1,I).NE.0.0D0) THEN
E=DABS(L(1,8)/Y(1,I))
ENDIF
E=DABS(L(1,8))
DO 84 K=1,X
IF(Y(K,I).NE.0.0D0) THEN
E=DMAX1(E,DABS(L(K,8)/Y(K,I)))
ENDIF
84 CONTINUE
ENDIF
E=DABS(E)
T(I+1)=T(I)+C(7)
IF(T(I+1).EQ.TEND) GO TO 30
LL(2)=E+1.0D-16
IF(LL(2)/LL(1).GT.100) THEN
IF(C(11).EQ.1.0D0) THEN
C(7)=0.0075D0
C(11)=C(11)+1.0D0
LL(1)=LL(2)
GO TO 20
ENDIF
C(11)=1.0D0
ENDIF
T(I+1)=T(I)+C(7)
C(7)=C(2)
I=I+1
GO TO 10
IF(C(12).LE.3.D0) THEN
  C(12)=C(12)+1.0D0
  LL(1)=LL(2)
  GO TO 20
ENDIF
  C(12)=1.0D0
  LL(1)=LL(2)
  GO TO 1
CONTINUE

FORMAT( //,'THE INTEGRATION ENDED IN SYSDEL WITH THE FOLLOWING VA
+LUES',
  + '/','TOL',2X,'=','2X,F15.12,
  + '/','TEND',2X,'=','2X,F10.5,
  + '/','THE NUMBER OF FUNCTION EVALUATIONS',3X,'=','F16.0,
  + '/','THE SOLUTION COMPONENTS ARE',
  + //,(1X,F24.16,/) )

C

  I=I+1
  WRITE(7,99) TOL,TEND,C(9),Y(K,I),K=1,X
  WRITE(7,*),Y(1,I)=DLOG(T(I)),Y(2,I)=1.0D0/T(I),T(I)
CALL CPTIME(1,110)
RETURN
END

SUBROUTINE NEWDIF(I,X1,X2,X3,X4,X5,X6,X7,T1,T2,T3,T4,T5,T6,T7,
  + D1,D2,D3,D4,D5,D6)
  IMPLICIT DOUBLE PRECISION(A-H,K-Z)

C

  THIS SUBROUTINE TAKES THE APPROPRIATE NUMBER OF VALUES OF THE SWITCHING
  FUNCTION AND CONSTRUCTS THE DIVIDED DIFFERENCE TABLE.

C

  D4=0.0D0
  D5=0.0D0
  D6=0.0D0
  D1=(X2-X1)/(T2-T1)
  A1=(X3-X2)/(T3-T2)
  B1=(X4-X3)/(T4-T3)
  D2=(A1-D1)/(T3-T1)
  A2=(B1-A1)/(T4-T2)
  D3=(A2-D2)/(T4-T1)
  IF(I.EQ.4) GO TO 1
  C1=(X5-X4)/(T5-T4)
  B2=(C1-B1)/(T5-T3)
  A3=(B2-A2)/(T5-T2)
  D4=(A3-D3)/(T5-T1)
IF(I.EQ.5) GO TO 1
E1=(X6-X5)/(T5-T6)
C2=(E1-C1)/(T6-T4)
B3=(C2-B2)/(T6-T3)
A4=(B3-A3)/(T6-T2)
D5=(A4-D4)/(T6-T1)
IF(I.EQ.6) GO TO 1
F1=(X7-X6)/(T7-T6)
E2=(F1-E1)/(T7-T5)
C3=(E2-C2)/(T7-T4)
B4=(C3-B3)/(T7-T3)
A5=(B4-A4)/(T7-T2)
D6=(A5-D5)/(T7-T1)
1 CONTINUE
RETURN
END

SUBROUTINE BISECT(D1,D2,D3,D4,D5,D6,E,T1,T2,T3,T4,T5,T6,R1)
IMPLICIT DOUBLE PRECISION(A-H,K-Z)

C
C THIS SUBROUTINE TAKES THE DIVIDED DIFFERENCE VALUES FROM THE SUBROUTINE
C NEWDIF AND EXTRAPOLATE THE SWITCHING FUNCTION F(S) BY USING THE
C NEWTON BACKWARD EXTRAPOLATION FORMULA OF THE APPROPRIATE DEGREE.
C E=F(T1)
C R: THE APPROXIMATION OF THE EXACT ROOT OF F(S) FOUND BY THE
C BISECTION METHOD.
C
F(S)=E+(S-T1)*D1+(S-T2)*D2+(S-T1)*D3
++(S-T1)*(S-T2)*D4
++(S-T1)*(S-T2)*(S-T3)*D5
++(S-T1)*(S-T2)*(S-T3)*(S-T4)*D6
A=T1-0.1D0
B=0.8D0*T1
1 U=F(A)
V=F(B)
IF(U*V)<2,7,7
2 DO 10 J=1,40
C=(A+B)*0.5
W=F(C)
IF(W*U)<3,7,4
3 B=C
V=W
GO TO 10

74
A=C
U=W
CONTINUE
CONTINUE
R1=C
RETURN
END

SUBROUTINE HERINT(I,J,S,Y1,Y2,Y3,T1,T2,T3,F1,F2,F3,V)
IMPLICIT DOUBLE PRECISION(A-E,K-Z)

THIS SUBROUTINE INTERPOLATES THE RETARTED SOLUTION Y(ALPHA(T,Y(T))
BY USING A THREE-POINT HERMITE INTERPOLATION FORMULA
S: THE POINT WHERE THE SOLUTION IS TO BE INTERPOLATED
V: THE APPROXIMATION OF Y(S) USING 3-POINTS HERM. INTERPO.

1994 FORMAT(//,1x,61(‘*’),/2(1x,’*’,59x,’*’),/1x,’*’,5x,
+’NUM. OF STEPS IS LESS THAN 3, PLEASE REDUCE C(2),6X,’*’.
+/,2(1x,’*’,59x,’*’),/1x,61(‘*’),//)
1993 FORMAT(//,1X,61(‘*’),/,2(1X,’*’,59X,’*’),/,1X,’*’,5X,
+10X,’TOL IS TOO LARGE, PLEASE REDUCE TOL’,6X,’*’,
+/,2(1X,’*’,59X,’*’),/1X,61(‘*’),//)
IF(J-2.GT.1) THEN
WRITE(5,1993)
STOP
ENDIF
IF(J.EQ.1.AND.I.LT.3) THEN
WRITE(5,1994)
STOP
ENDIF
H1=((S-T2)*(S-T3))/((T1-T2)*(T1-T3))
H2=((S-T1)*(S-T3))/((T2-T1)*(T2-T3))
H3=((S-T1)*(S-T2))/((T3-T1)*(T3-T2))
H11=((T1-T3)+(T1-T2))/((T1-T2)*(T1-T3))
H21=((T2-T1)+(T2-T3))/((T2-T1)*(T2-T3))
H31=((T3-T1)+(T3-T2))/((T3-T1)*(T3-T2))
V=(1-2*H11*(S-T1))*H1**2*Y1
+ (1-2*H21*(S-T2))*H2**2*Y2
+ (1-2*H31*(S-T3))*H3**2*Y3
+ (S-T1)*H1**2*F1+(S-T2)*H2**2*F2+(S-T3)*H3**2*F3
RETURN
END
SUBROUTINE EVAL(I,K,X,M,D,B,T,Y,F,V,U,J)
IMPLICIT DOUBLE PRECISION(A-H,L-Z)
INTEGER X,XM
DIMENSION T(XM),Y(X,XM),F(X,XM),V(X),B(X)

C
C THIS SUBROUTINE LOCATES THE DELAYED TERM ALPHA(T,Y(T)) BETWEEN TWO
C PREVIOUS GRID POINTS, THEN IT INTERPOLATES THE SOLUTION AT THE
C DELAYED TERM BY USING A 3-POINT HERMITE INTERPOLATION.
C
TT=T(1)
AA= AL(K,D,B)
IF(AA.LE.TT) THEN
   DO 112 II=1,X
   V(II)=PHI(K,AA)
   AA= AL(K,D,B)
112 CONTINUE
ENDIF
U= G(K,D,B,V)
IF(AA.GT.TT) THEN
   BETTA1=AA
   IF(BETTA1-TT)447,447,448
447 J=1
   GO TO 4524
448 J=2
   GO TO 4521
4521 IF(BETTA1-T(J))4523,4523,4522
4523 J=J-1
   GO TO 4524
4522 J=J+1
   GO TO 4521
4524 CONTINUE
   S=BETTA1
   DO 99 II=1,X
   CALL HERINT(I,J,S,Y(II,J),Y(II,J+1),Y''(I,J+2),T(J),T(J+1),T(J+2)
   +,F(II,J),F(II,J+1),F(II,J+2),V(II))
99 CONTINUE
ENDIF
U= G(K,D,B,V)
RETURN
END

C
C
C
C
C***
SUBROUTINE CPTIME(ISTART, IFRMAT)

GIVES THE CPU TIME IN SECONDS USED DURING EXECUTION

ISTART = 0 : SET THE CLOCK
    = 1 : GET THE CPU TIME

IF(IFRAMAT.EQ.100) WRITE(5,100) RCPU-RCPU1
IF(IFRAMAT.EQ.110) WRITE(5,110) RCPU-RCPU1
IF(IFRAMAT.EQ.120) WRITE(5,120) RCPU-RCPU1
IF(IFRAMAT.EQ.999) WRITE(5,999) RCPU-RCPU1

100 FORMAT(''/,1X,61(''*'),/,'*,59X,'**',/,'1X,'**',5X,  
    'TIMER SET TO ZERO AT THE BEGINNING OF THE PROGRAM',5X,'**',  
    '/,2(1X,'**',59X,'**',/),1X,61('**'),/*)')
110 FORMAT(''/,1X,61(''''),/,2(1X,'''',59X,'''',/),1X,'''',5X,  
    'TIME BEFORE CALL TO MAIN = ',F10.4,' CPU SECONDS',2X,'''',  
    '/,2(1X,'''',59X,'''',/),1X,61(''''),/*)')
120 FORMAT(''/,1X,61(''''),/,2(1X,'''',59X,'''',/),1X,'''',5X,  
    'TIME AFTER CALL TO MAIN = ',F10.4,' CPU SECONDS',2X,'''',  
    '/,2(1X,'''',59X,'''',/),1X,61(''''),/*)')
999 FORMAT(''/,1X,61(''''),/,2(1X,'''',59X,'''',/),1X,'''',5X,  
    'TIME AT THE END OF PROGRAM = ',F10.4,' CPU SECONDS',2X,'''',  
    '/,2(1X,'''',59X,'''',/),1X,61(''''),/*)')
RETURN
END