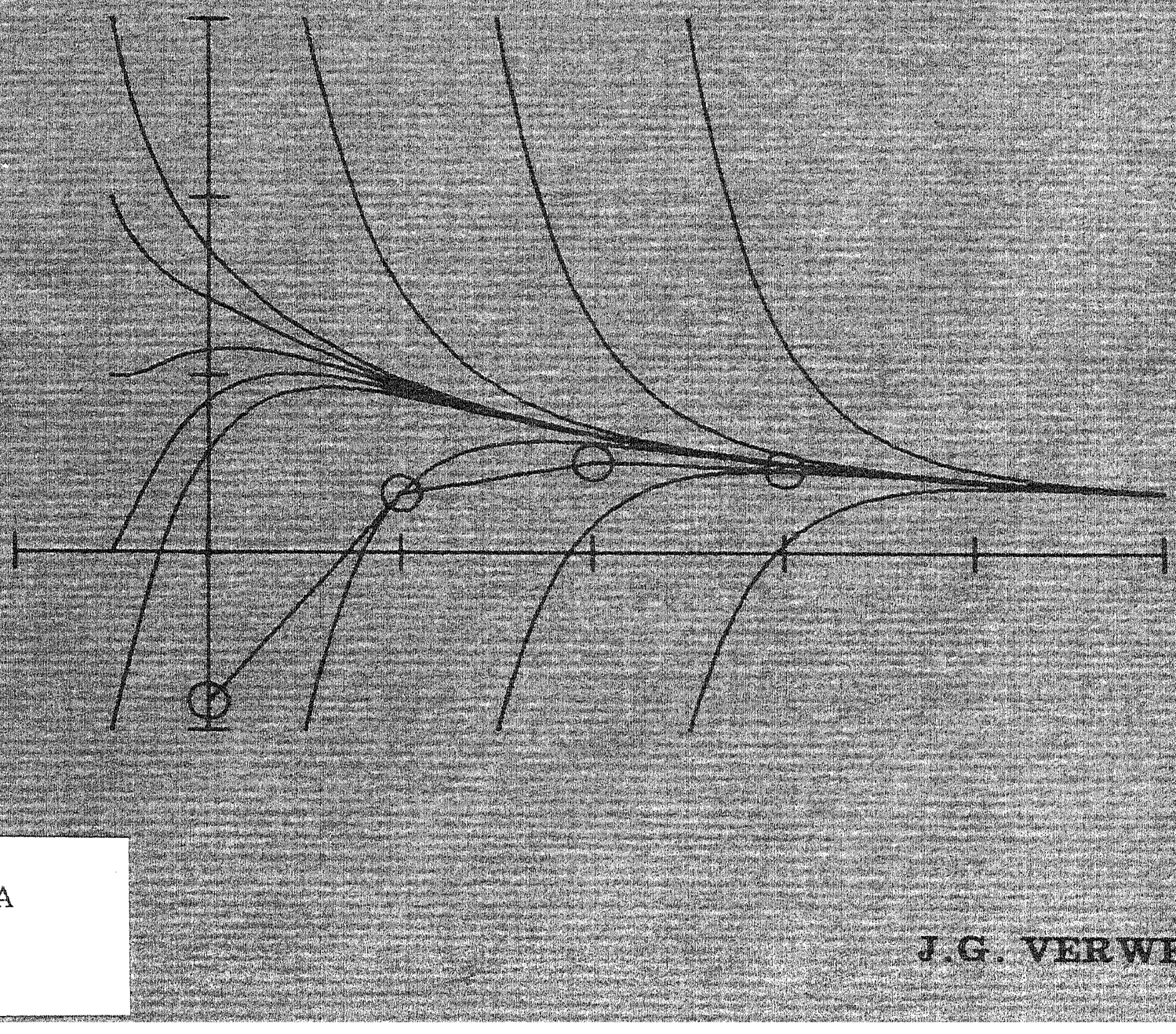
ON THE CONSTRUCTION AND ANALYSIS OF STABLE NUMERICAL METHODS FOR STIFF AND PARABOLIC DIFFERENTIAL EQUATIONS



J.G. VERWER

ON THE CONSTRUCTION AND ANALYSIS OF STABLE NUMERICAL METHODS FOR STIFF AND PARABOLIC DIFFERENTIAL EQUATIONS

ON THE CONSTRUCTION AND ANALYSIS OF STABLE NUMERICAL METHODS FOR STIFF AND PARABOLIC DIFFERENTIAL EQUATIONS

ACADEMISCH PROEFSCHRIFT

TER VERKRIJGING VAN DE GRAAD VAN
DOCTOR IN DE WISKUNDE EN NATUURWETENSCHAPPEN
AAN DE UNIVERSITEIT VAN AMSTERDAM,
OP GEZAG VAN DE RECTOR MAGNIFICUS
DR G. DEN BOEF,
HOOGLERAAR IN DE FACULTEIT DER WISKUNDE EN NATUURWETENSCHAPPEN,
IN HET OPENBAAR TE VERDEDIGEN
IN DE AULA DER UNIVERSITEIT
(TIJDELIJK IN DE LUTHERSE KERK, INGANG SINGEL 411, HOEK SPUI)
OP WOENSDAG 23 NOVEMBER 1977 DES NAMIDDAGS TE 3.00 UUR PRECIES

DOOR

JOHANNES GERARDUS VERWER

GEBOREN TE HEERHUGOWAARD

1977
MATHEMATISCH CENTRUM, AMSTERDAM

PROMOTOR : PROF.DR. P.J. VAN DER HOUWEN

COPROMOTOR: PROF.DR. T.J. DEKKER

COREFERENT: PROF.DR. A. VAN DER SLUIS

The figure on the front has been copied from:
Dekker, T.J. & P.W. Hemker, P.J. van der Houwen,
Colloquium Stijve Differentiaalvergelijkingen, Deel 1,
MC Syllabus 15.1, Mathematisch Centrum, Amsterdam, 1972.

PREFACE

My thesis consists of four papers on the numerical integration of initial value problems for two related classes of ordinary differential equations, viz. stiff equations and semi-discretized parabolic equations. Two papers have been published in *Numerische Mathematik* and one paper will appear in the *Journal of Computational and Applied Mathematics*. The fourth paper has been submitted for publication. A prepublication of this paper has appeared in the *NW-series of the Mathematical Centre*. The papers are:

On generalized linear multistep methods with zero-parasitic roots and an adaptive principal root, Numerische Mathematik 27, 143-155, 1977.

S-stability properties for generalized Runge-Kutta methods, Numerische Mathematik 27, 359-370, 1977.

A class of stabilized three-step Runge-Kutta methods for the numerical integration of parabolic equations, Journal of Computational and Applied Mathematics, (to appear).

An implementation of a class of stabilized, explicit methods for the time integration of parabolic equations, Report NW 38/77, Mathematisch Centrum, Amsterdam, (prepublication), 1977.

Separate copies of these papers may be obtained from the Mathematical Centre.

In order to facilitate the reading for those who are unexperienced in this area, I wrote a small introductory part. In this part the difficulties encountered when solving stiff and parabolic problems are illustrated. In addition, the four papers are related with each other.

I should like to express my gratitude to the governing board of the "Stichting Mathematisch Centrum" for giving me the opportunity to carry out the investigations, and for publishing my thesis. In particular, I want to express my gratitude to Prof.Dr. P.J. van der Houwen for introducing me to the field of numerical integration and for his stimulating

guidance as a promotor. I am also grateful to Prof.Dr. T.J. Dekker and Prof.Dr. A. van der Sluis for their critical and inspiring remarks and for their willingness to be my copromotor and "coreferent", respectively.

I want to thank the members of the Working Group on Differential and Integral Equations of the Department of Numerical Mathematics of the Mathematical Centre for the stimulating and instructive discussions on each other's research. In particular, I want to thank Mr. C. den Heyer for his careful reading of early manuscripts of parts of my thesis, and Mr. B.P. Sommeyer for his programming assistance during the last year.

Finally, I wish to acknowledge Dr. J. van de Lune and Mrs. S.G. Greene for correcting parts of the English text, Mrs. R.W.T. Riechelmann-Huis for the typing of the manuscripts and Mr. D. Zwarst for the printing and the binding.

J.G. Verwer
Mathematisch Centrum
2e Boerhaavestraat 49
Amsterdam

CONTENTS

THE INTRODUCTORY PART

1. INTRODUCTION

The aforementioned papers deal with numerical integration of initial value problems for systems of ordinary differential equations of the type

$$\frac{d\vec{y}}{dx} = \vec{f}(x,\vec{y}).$$

Two of these papers, viz. VERWER [21,22], discuss the numerical integration of stiff equations, while the other two, viz. VERWER [23,24], are devoted to the numerical integration of large systems originating from semi-discretization of parabolic initial boundary value problems. These two classes of differential equations join the property of imposing severe stability requirements on the difference schemes used to produce a numerical solution. In three of the four papers the emphasis is on the *construction* and *analysis* of difference schemes which meet these *stability* requirements. The fourth one discusses a computer *implementation* of one of those schemes.

2. METHODS FOR STIFF DIFFERENTIAL EQUATIONS

Many physical processes give rise to systems of ordinary differential equations whose solutions contain slowly as well as rapidly varying components, the latter dying out relatively fast. Differential systems exhibiting this phenomenon have become known as stiff systems, a terminology introduced by CURTISS & HIRSCHFELDER [4]. A precise mathematical definition of stiffness does not seem to exist in the literature. We use the following, somewhat heuristic

DEFINITION. A system of differential equations is said to be stiff at the point $(x_0,\vec{y}_0)$, if all eigenvalues of $\partial\vec{f}/\partial\vec{y}$, evaluated at $(x_0,\vec{y}_0)$, have negative real parts and are located in two or more widely separated clusters, at least one of them lying close to the origin.

A more precise definition of stiffness is given in VAN DER HOUWEN [18, p.10]. It is of importance to observe that, according to these definitions, it is required that the eigenvalues are located in widely separated clusters. On the other hand, LAMBERT [10, p.231] defines a system as being

stiff if all eigenvalues have negative real parts which differ greatly in magnitude. Thus, according to the definition given by Lambert, a system of which the eigenvalues are more or less equally spaced may also be called a stiff system. An example of such a system is provided by a semi-discretized parabolic equation (see section 3). The general solution of this type of equation contains rapidly as well as slowly varying components. However, no sharp distinction between these components can be made. Therefore, we restrict the definition of stiffness to systems of which the eigenvalues are widely separated.

When the numerical integration process of a stiff system has proceeded to a point where the rapidly varying components sufficiently died out, one would like to take stepsizes which are related to the rate of variation of the slow components. This is not permitted however by standard techniques, such as explicit Runge-Kutta methods, for which it is always necessary to integrate with stepsizes taking account of the rapid components. To illustrate this point we consider a typical stiff problem being used by many authors to test their algorithms. This non-linear problem originates from the field of chemical kinetics (ROBERTSON [16]):

$$\frac{dy_1}{dx} = -0.04\, y_1 + 10^4\, y_2 y_3, \quad y_1(0) = 1,$$

$$\frac{dy_2}{dx} = 0.04\, y_1 - 10^4\, y_2 y_3 - 3.10^7\, y_2^2, \quad y_2(0) = 0,$$

$$\frac{dy_3}{dx} = 3.10^7\, y_2^2, \quad y_3(0) = 0.$$

This system represents a set of chemical reaction rate equations which are dependent ($y_1 + y_2 + y_3 = 1$). The Jacobian matrix is

$$\begin{pmatrix} -0.04 & 10^4 y_3 & 10^4 y_2 \\ 0.04 & -10^4 y_3 - 6.10^7 y_2 & -10^4 y_2 \\ 0 & 6.10^7 y_2 & 0 \end{pmatrix}$$

Its eigenvalues are given by $\delta_1 = 0$ and

$$\delta^2 + (0.04 + 10^4 y_3 + 6.10^7 y_2)\ \delta + (24.10^5 y_2 + 6.10^{11} y_2^2) = 0.$$

The roots of this equation, say δ_2 and δ_3, are real along the solution curve. At $x = 0$, $\delta_2 = 0$ and $\delta_3 = -0.04$. As soon as $x > 0$, δ_3 decreases rapidly to the asymptotic value -10^4, whereas δ_2 remains in the neighbourhood of the origin. When integrating this problem with the standard third order Runge-Kutta formula of Heun (LAMBERT [10, p.119]), it is necessary to choose the stepsize smaller than or equal to $2.51/10^4$. Here, the number 2.51 represents the real boundary of absolute stability of the Heun formula. If the stepsize does not satisfy this stability restriction the numerical solution is "blown up" immediately. The total interval of interest for the present problem is $0 \leq x \leq 40$, so that roughly 160.000 integration steps, or 480.000 function evaluations, would be necessary, requiring an excessive amount of computing time. Since the restriction on the stepsize is not due to accuracy requirements (see e.g. the experiments reported by VAN KAMPEN [20]), it is clear that special techniques are needed.

In order to cope with the stability requirements posed by the large eigenvalues an efficient method for solving stiff systems always makes use of the Jacobian matrix. Accordingly, methods for stiff equations can be roughly divided into two categories::

A) *Implicit methods* applied in conjunction with a Newton-Raphson type iteration. Here the Jacobian matrix, or an approximation to it, is used in the iteration. Examples of well-known implicit methods for stiff problems are the linear multistep backward differentiation methods which were proposed and implemented by GEAR [8], and the Runge-Kutta methods discussed by BUTCHER [3].

B) *Generalized methods*, i.e. methods where the Jacobian matrix is introduced directly into the coefficients of the integration formula (this idea goes back to ROSENBROCK [17]). Any implicit method applied with a fixed number of Newton-Raphson iterations is in fact a generalized method (see VAN DER HOUWEN [18, e.g. section 2.3]).

In the first and second paper of the thesis two classes of generalized methods are discussed, viz. generalized linear multistep methods and generalized Runge-Kutta methods. These methods have in common that the matrix coefficients of the integration formulas are generated by rational functions. These rational functions are chosen in order to obtain a certain order of accuracy, and to obtain favourable stability properties, such as A-stability (DAHLQUIST [5]). The methods are explicit. However, due to the presence of the rational functions, they require the solution of systems of linear algebraic equations. Because of this they are also called semi-implicit or linearly implicit (LAMBERT [10, p.246]).

In the first paper the emphasis is on the stability behaviour of a class of multistep methods with zero-parasitic roots and an adaptive principal root (= stability function). This means that the stability behaviour of such a method can be adapted to the problem under consideration. As a consequence, such a method may be suitable for semi-discretized parabolic and hyperbolic differential equations, which also impose severe stability requirements on corresponding difference schemes (see e.g. RICHTMYER & MORTON [15]). In the second paper the emphasis is on the stability behaviour of generalized Runge-Kutta methods when applied to stiff, non-linear problems. A-stable methods from this class often fail when applied to such problems. It is shown that for these problems S-stable methods are superior to A-stable ones, and that S-stable methods are suited for strongly non-linear, highly stiff problems. The concept of S-stability has been proposed by PROTHERO & ROBINSON [14] for implicit Runge-Kutta methods. It is related to a test-equation which gives a better indication of the stability behaviour of a Runge-Kutta type method for a stiff, non-linear problem, compared to the usual test-equation related to the concept of A-stability.

We conclude this section by giving some more references. The first paper on stiff equations is the already mentioned 1952 paper by Curtiss and Hirschfelder. They clearly pointed out the need for special methods for stiff problems, especially in the field of chemical kinetics. After this first paper, many have been published on the construction and analysis of efficient methods for existing as well as new problem fields. The majority of these papers were written after the 1968 IFIP Congress in Edinburgh

(MORRELL [12]). After this congress it was recognized throughout the numerical analysis community that the integration of stiff equations was an important and difficult problem. A somewhat obsolete, but excellent survey of methods and problems is given by BJUREL et al [2]. This report appeared in 1970 and gives several fields of applications, e.g. dynamics, control theory, reactor kinetics and chemical kinetics. Of a more recent date are the Proceedings of the International Symposium on Stiff Differential Systems, held at Wildbad in Germany in 1973 (WILLOUGHBY [26]). Besides contributions to applications this book also contains theoretical work. Textbooks paying special attention to stiff equations are, in chronological order, LAPIDUS & SEINFELD [11], GEAR [8], LAMBERT [10] and VAN DER HOUWEN [18].

3. METHODS FOR SEMI-DISCRETIZED PARABOLIC DIFFERENTIAL EQUATIONS

A flexible method in the numerical solution of parabolic and hyperbolic differential equations is *the method of lines*, a method originally proposed by Russian mathematicians (BEREZIN & ZHIDKOV [1]). By applying this method the numerical solution process consists of two parts, viz. semi-discretization and time-integration.

Semi-discretization. The partial differential equation is converted into a (generally large) system of ordinary differential equations by discretizing the space variables, while the time variable is left continuous. Semi-discretization is usually based on two discretization methods; either the finite element method (DOUGLAS & DUPONT [7]), or the finite difference method (RICHTMYER & MORTON [15]). An important property of the majority of semi-discretized parabolic equations is that the eigenvalues of the Jacobian matrix are lying in a long narrow strip around the negative axis of the complex plane. To illustrate this important property we consider the simple, one-dimensional heat flow problem

$$u_t = u_{xx}, \quad t > 0, \quad 0 \le x \le 1,$$

$$u(0,x) = \varphi(x), \quad 0 \le x \le 1,$$

$$u(t,0) = u(t,1) = 0, \quad t > 0.$$

For any given N let $x_i = i\,\Delta x$, for $i = 0,\ldots,N+1$, where $\Delta x = (N+1)^{-1}$. Let $u_i(t)$ denote the approximating function for $u(t,x_i)$, which is defined by replacing u_{xx} at (t,x_i) by the standard finite difference quotient

$$\frac{u(t,x_i-\Delta x)-2u(t,x_i)+u(t,x_i+\Delta x)}{(\Delta x)^2}\,.$$

The functions $u_i(t)$, $i = 1,\ldots,N$, then satisfy the initial value problem

$$\frac{d\vec{u}}{dt} = J\vec{u}, \quad \vec{u}(0) = \vec{\varphi},$$

where $\vec{u} = [u_1,\ldots,u_N]^T$, $\vec{\varphi} = [\varphi(x_1),\ldots,\varphi(x_N)]^T$, and

$$J = (\Delta x)^{-2}\begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix}.$$

The eigenvalues, say δ_i, of the symmetric Jacobian matrix J are given by (ORTEGA & RHEINBOLDT [13, p.51])

$$\delta_i = (N+1)^2[-2 + 2\cos(\tfrac{i\pi}{N+1})], \quad i = 1,\ldots,N.$$

We see that the eigenvalues are distributed over the interval $[-4(N+1)^2,0]$, and (because N is usually large) *differ greatly in magnitude*. Therefore, these equations are sometimes called stiff (see the discussion in section 2). Though the present problem is rather simple, it clearly illustrates that semi-discretized parabolic equations, as well as stiff equations, can not be efficiently handled by means of standard explicit integration formulas.

Time-integration. The resulting system of ordinary differential equations is solved by applying a numerical integration method. Time-integration methods for semi-discretized partial differential equations might be divided into three categories:

A) *Explicit methods*, such as classical Runge-Kutta methods.
B) *Implicit methods*, such as linear multistep methods of the backward differentiation type (GEAR [8]), and *generalized methods* (see section 2).
C) *Splitting methods*, such as the method of alternating directions, the locally one-dimensional method, and the hopscotch method (see e.g. YANENKO [27] and GOURLAY [9]).

Each category has its advantages and disadvantages. A clear advantages of category A is that an explicit method is easy to apply and very robust, i.e. applicable to a very general class of problems. A clear disadvantage of an explicit method is that such a method necessarily possesses conditional stability properties, in contrast with methods from category B, which may be unconditionally stable. The disadvantage of the latter methods is that for multi-dimensional problems, large linear algebraic systems with a complicated structure must be solved. The solution of these systems is generally very time consuming and can easily annul the advantage of unrestricted stepsizes. To circumvent these time consuming solutions, splitting methods have been introduced. The idea of splitting is to split a complicated process into a series of simple processes. The advantage of this approach is that for relevant classes of problems an efficient process is obtained, while unconditional stability is preserved. A clear disadvantage of splitting methods, when compared to methods from category A and B, is that these methods are not so easy to apply and are less robust.

We observe that all methods belonging to the three categories are also applied as *direct grid methods*, i.e. methods which simultaneously discretize the time and space variables of the partial differential equation. In particular, splitting methods are mostly applied in this way. In our opinion, it is preferable to apply the method of lines, since this leads to a more systematic treatment of methods for time dependent partial differential equations (see e.g. VAN DER HOUWEN & VERWER [19]).

In the third paper of the thesis we propose a class of explicit Runge-Kutta formulas for the time-integration of semi-discretized parabolic equations. These formulas thus belong to category A mentioned above. In the paper the discussion is concentrated on the construction and analysis of formulas with extended real stability intervals, in order to reduce the disadvantage of conditional stability properties. Stabilized formulas of order $p = 1$ and $p = 2$ are presented, possessing real stability boundaries approximately equal to $(2.29 + (2 - p)\ 2.86)\ m^2$, m denoting the number of function evaluations per integration step. This number may vary between 2 and 12. For comparison, these stability boundaries are nearly three times as large as the stability boundaries of similar stabilized formulas given by VAN DER HOUWEN [18, section 2.6.6]. In spite of possessing finite stability regions, the advantage of easy applicability and robustness makes a stabilized, explicit method suited for semi-discretized parabolic equations.

The fourth paper of the thesis is a continuation of the third one. In this fourth paper we discuss a computer implementation of a class of stabilized, explicit Runge-Kutta formulas. The implementation is provided with stepsize and error control. A commented and portable FORTRAN program based on this implementation is added. Thanks to various control mechanisms this program performs the time-integration automatically.

REFERENCES

[1] BEREZIN, I.S. & N.P. ZHIDKOV, *Computing methods II*, Pergamon Press, Oxford, 1965.

[2] BJUREL, G. & G. DAHLQUIST, B. LINDBERG, S. LINDE, L. ODEN, *Survey of stiff ordinary differential equations*, Comp. Sc. Report NA 70.11, Royal Inst. of Techn., Dept. of Inf. Processing, Stockholm, 1970.

[3] BUTCHER, J.C., *Implicit Runge-Kutta processes*, Math. Comp. 18, 50-64, 1964.

[4] CURTISS, C.F. & J.O. HIRSCHFELDER, *Integration of stiff equations*, Proc. Nat. Acad. Sci. U.S. 38, 235-243, 1952.

[5] DAHLQUIST, G., *A special stability problem for linear multistep methods*, BIT 3, 27-43, 1963.

[6] DEKKER, K. & P.J. VAN DER HOUWEN, J.G. VERWER, P.H.M. WOLKENFELT, *Comparing stabilized Runge-Kutta methods for semi-discretized parabolic and hyperbolic equations*, Report NW 45/77, Mathematisch Centrum, Amsterdam, 1977.

[7] DOUGLAS, J. & T. DUPONT, *Galerkin methods for parabolic equations*, SIAM J. Numer. Anal. 7, 575-626, 1970.

[8] GEAR, C.W., *Numerical initial value problems in ordinary differential equations*, Prentice Hall, Englewood Cliffs, New Jersey, 1971.

[9] GOURLAY, A.R., *Splitting methods for time dependent partial differential equations*, Proceedings of the 1976 conference: *The state of the art in numerical analysis*, ed. D.A.H. Jacobs, Academic Press (to appear).

[10] LAMBERT, J.D., *Computational methods in ordinary differential equations*, John Wiley & Sons, London, 1973.

[11] LAPIDUS, L. & J.H. SEINFELD, *Numerical solution of ordinary differential equations*, Academic Press, New York, 1971.

[12] MORRELL, A.J.H., ed., *IFIP Conf. Edinburgh*, North-Holland, Amsterdam, 1968.

[13] ORTEGA, J.M. & W.C. RHEINBOLDT, *Iterative solution of nonlinear equations in several variables*, Academic Press, New York and London, 1970.

[14] PROTHERO, A. & A. ROBINSON, *On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations*, Math. Comp. 28, 145-162, 1974.

[15] RICHTMYER, R.D. & K.W. MORTON, *Difference methods for initial value problems*, Interscience, New York, 1967.

[16] ROBERTSON, H.H., *The solution of a set of reaction rate equations in numerical analysis*, ed. J. Walsh, Thompson Book Co., Washington, 1967.

[17] ROSENBROCK, H.H., *Some general implicit processes for the numerical solution of differential equations*, Computer J. 5, 329-330, 1963.

[18] VAN DER HOUWEN, P.J., *Construction of integration formulas for initial value problems*, North-Holland Publishing Company, Amsterdam, 1976.

[19] VAN DER HOUWEN, P.J. & J.G. VERWER, *A general formulation of linear splitting methods for ordinary and partial differential equations*, Report NW 47/77, Mathematisch Centrum, Amsterdam, 1977.

[20] VAN KAMPEN, S.P.N., *Colloquium stijve differentiaalvergelijkingen*, deel 2, hoofdstuk 8, (Dutch), Mathematisch Centrum, Amsterdam, 1973.

[21] VERWER, J.G., *On generalized linear multistep methods with zero-parasitic roots and an adaptive principal root*, Numer. Math. 27, 143-155, 1977.

[22] VERWER, J.G., *S-stability properties for generalized Runge-Kutta methods*, Numer. Math. 27, 359-370, 1977.

[23] VERWER, J.G., *A class of stabilized three-step Runge-Kutta methods for the numerical integration of parabolic equations*, J. of Computational and Applied Math., (to appear).

[24] VERWER, J.G., *An implementation of a class of stabilized, explicit methods for the time integration of parabolic equations*, Report NW 38/77, Mathematisch Centrum, Amsterdam, (prepublication), 1977.

[25] VERWER, J.G., *A comparison between the odd-even hopscotch method and a class of Runge-Kutta methods with extended real stability intervals*, Report NN 13/77, Mathematisch Centrum, Amsterdam, 1977.

[26] WILLOUGHBY, R.A., (ed), *Stiff differential systems*, Plenum Press, New York and London, 1974.

[27] YANENKO, N.N., *The method of fractional steps*, Springer Verlag, Berlin, 1971.

THE FOUR PAPERS

Numer. Math. 27, 143—155 (1977)

On Generalized Linear Multistep Methods with Zero-Parasitic Roots and an Adaptive Principal Root

J. G. Verwer

Received May 27, 1975

Summary. A class of linear multistep methods is considered for the numerical integration of stiff systems of ordinary differential equations. These methods are characterized by the fact that the coefficients of the integration formulas are matrices depending on the Jacobian or on an approximation to the Jacobian. They have the possibility to adapt the characteristic root of the method to the problem under consideration. Special attention is paid to stability aspects. Numerical results are reported.

1. Introduction

Let

$$y' = f(x, y),\ y(x_0) = y_0, \tag{1.1}$$

represent an initial value problem for a set of ordinary differential equations of which the real vector function $f(x, y)$ is sufficiently differentiable. Let $J(x, y)$ denote the Jacobian matrix $\partial f/\partial y$ and let A_l and B_l, $l=1, \ldots, k$, denote rational functions with real coefficients. In order to solve (1.1) we consider linear multistep formulas of the type (cf. Van der Houwen and Verwer [14])

$$y_{n+1} = \sum_{l=1}^{k} A_l(h_n J_n)\, y_{n+1-l} + h_n \sum_{l=1}^{k} B_l(h_n J_n)\, f_{n+1-l}, \tag{1.2}$$

where J_n is a matrix which equals or approximates $J(x_n, y_n)$.

Such a generalized multistep method bears a resemblance to implicit linear multistep methods with scalar coefficients, when used in conjunction with a Newton type method with one iteration per integration step. The idea to introduce the Jacobian directly into the coefficients of the integration formula proceeds from Rosenbrock [9].

Because we assume that the functions A_l and B_l are rational, formula (1.2) is not fully explicit. To obtain y_{n+1} it is necessary to invert one or more matrices. Lambert and Sigurdsson [3] discuss a class of multistep methods with variable matrix coefficients, which includes fully implicit formulas.

Let us apply scheme (1.2) to the stability test-equation

$$y' = Jy, \tag{1.3}$$

where J is a constant matrix, and assume $J_n = J$. Then, (1.2) is reduced to the

k-step difference equation

$$y_{n+1}=\sum_{l=1}^{k}\left[A_l(h_n J)+h_n J B_l(h_n J)\right] y_{n+1-l}. \tag{1.4}$$

Further, let R be any rational function and suppose

$$\begin{aligned} R(z)&=A_1(z)+zB_1(z),\\ A_l(z)+zB_l(z)&=0, \qquad l=2,\ldots,k. \end{aligned} \tag{1.5}$$

Substitution of relations (1.5) into equation (1.4) yields the one-step difference equation

$$y_{n+1}=R(h_n J)\, y_n. \tag{1.6}$$

Thus, by an appropriate choice of the rational functions in (1.2), we arrive at a multistep formula with zero-parasitic roots and an adaptive principal root, i.e., a formula of which the stability function may be identified with a prescribed rational function R. The formula reads

$$y_{n+1}=R(h_n J_n)\, y_n+h_n\sum_{l=1}^{k} B_l(h_n J_n)\left[f_{n+1-l}-J_n y_{n+1-l}\right]. \tag{1.7}$$

We remark, however, that using approximations to the Jacobian matrix involves that the stability function of (1.7) is only approximately given by R and the parasitic roots are only approximately zero.

Because of the adaptivity, formula (1.7) can be used for efficient integration of parabolic, hyperbolic and stiff differential equations. In this paper we mainly discuss method (1.7) when applied to stiff equations. In section 2 we derive expressions for the functions B_l emanating from arbitrary approximations to the exponential. Section 3 deals with the stability behaviour of method (1.7) in case of inaccurately evaluated Jacobians, i.e., $J_n \neq J(x_n, y_n)$. In section 4 we discuss the S-stability and stiff-consistency properties (cf. Prothero and Robinson [7]) of method (1.7). In the last section some numerical results are presented.

2. Consistency Conditions

For generalized schemes, such as (1.2), we can derive two types of consistency conditions. The first type requires an accurate Jacobian matrix, i.e. the Jacobian has to be calculated within some fixed order of accuracy, while the second type allows arbitrary approximations to the Jacobian. In order to reduce computational effort, we prefer consistency conditions of the second type. Formulas requiring accurate Jacobian matrices are discussed in Van der Houwen [12, 13].

Let x_j, $j=n+1, n, \ldots, n+1-k$, denote the reference points of the k-step formula, and let $h_n=x_{n+1}-x_n$ denote the steplength. Further, let y_n denote the numerical approximation to the analytical solution $y(x)$, of problem (1.1), at $x=x_n$, and let $f_n=f(x_n, y_n)$. Define the numbers $q_l=(x_{n-l}-x_n)/h_n$, $l=-1,0,1,\ldots,k-1$. We associate scheme (1.7) with the linear difference operator $\mathscr{C}$ defined by

$$\begin{aligned} &\mathscr{C}[y(x_n); h_n; J_n]=\\ &y(x_{n+1})-R(h_n J_n)\, y(x_n)-h_n\sum_{l=1}^{k} B_l(h_n J_n)\left[y'(x_{n+1-l})-J_n y(x_{n+1-l})\right]. \end{aligned} \tag{2.1}$$

For the purpose of our analysis we assume that the matrices $R(h_n J_n)$ and $B_l(h_n J_n)$ are non-singular.

Definition 2.1. Let y be a solution of the differential equation. Then scheme (1.7) is said to be consistent of order p at $x=x_n$, when for any matrix J_n and any set of fixed numbers q_l, $l=1, \ldots, k-1$,

$$\mathscr{C}[y(x_n); h_n; J_n] = O(h_n^{p+1}) \text{ as } h_n \to 0. \tag{2.2}$$

Let $y^{(j)}(x_n)$ denote the j-th derivative of y at $x=x_n$. By expanding $y(x_{n+1-l})$ and $y'(x_{n+1-l})$ about x_n, we may formally derive

$$\mathscr{C}[y(x_n); h_n; J_n] = \sum_{j=0}^{\infty} \left[\frac{1}{j!} - C_j(h_n J_n)\right] h_n^j y^{(j)}(x_n), \tag{2.3}$$

where the rational functions C_j are defined by

$$\begin{aligned} C_0(z) &= R(z) - \sum_{l=1}^{k} z B_l(z), \\ C_j(z) &= \frac{1}{j!} \sum_{l=1}^{k} (j - z\, q_{l-1})\, q_{l-1}^{j-1}\, B_l(z), \qquad j=1, 2, \ldots. \end{aligned} \tag{2.4}$$

According to Definition 2.1, the consistency conditions for p-th order accuracy are now easily seen to be

$$C_j(z) = \frac{1}{j!} + O(z^{p+1-j}) \text{ as } z \to 0, \qquad j=0, \ldots, p. \tag{2.5}$$

To obtain expressions for the functions B_l, we substitute expressions (2.4) into conditions (2.5). Then, after some calculations, we arrive at the Vandermonde system (see Van der Houwen and Verwer [14])

$$\sum_{l=1}^{k} q_{l-1}^{j-1} B_l(z) = D_j(z), \qquad j=1, \ldots, p+1, \tag{2.6}$$

where the functions D_j are defined by

$$\begin{aligned} D_1(z) &= \frac{R(z) - 1 + \varepsilon_{p+1}(z)}{z}, \\ D_{j+1}(z) &= \frac{j D_j(z) - 1 - \varepsilon_{p+1-j}(z)}{z}, \qquad j=1, \ldots, p. \end{aligned} \tag{2.7}$$

The rational functions ε_j are related to the order terms in (2.5) and may be freely chosen provided $\varepsilon_j(z) = O(z^j)$ as $z \to 0$.

According to Definition 2.1, J_n may be put equal to the zero matrix. For this choice scheme (1.7) is reduced to a scheme of the classical Adams-Bashforth type. This implies that its order $p \leqq k$. We shall show that system (2.6) has a solution for $p=k$.

The function R may play the role of the stability function. Therefore it has to be a consistent approximation to the exponential.

Definition 2.2. The rational function R is consistent of order p if

$$\frac{d^i}{dz^i} R(z)\Big|_{z=0} = 1, \qquad i=0, \ldots, p. \tag{2.8}$$

Relation (2.8) implies that $R(z)=e^z+O(z^{p+1})$ as $z\to 0$.

Theorem 2.1. Let R be consistent of order $\geqq k$ and let the functions B_l, $l=1, \ldots, k$, be determined by relations

$$\begin{aligned} \sum_{l=1}^{k} q_{l-1}^{j-1} B_l(z) &= D_j(z), \qquad j=1, \ldots, k, \\ D_1(z) &= \frac{R(z)-1+\varepsilon_{k+1}(z)}{z}, \\ D_{j+1}(z) &= \frac{j D_j(z)-1-\varepsilon_{k+1-j}(z)}{z}, \qquad j=1, \ldots, k-1, \end{aligned} \tag{2.9}$$

then, the order p of scheme (1.7) equals k.

Proof. It is sufficient to show that the $(p+1)\times k$ Vandermonde system (2.6) has a solution for $p=k$. The recurrence relation (2.7) yields

$$D_j(z) = (j-1)!\,\frac{R(z)-P_{j-1}(z)}{z^j} + O(z^{p-j+1}), \; z\to 0, \qquad j=1, \ldots, p+1, \tag{2.10}$$

where

$$P_{j-1}(z) = 1+z+\cdots+\frac{z^{j-1}}{(j-1)!}. \tag{2.11}$$

Let $p=k$. As R is consistent of order greater or equal than k, the asymptotic relation (2.10) may be written as

$$D_j(z) = (j-1)!\sum_{i=0}^{k-j}\frac{z^i}{(j+i)!} + O(z^{k+1-j}), \; z\to 0, \qquad j=1, \ldots, k+1. \tag{2.12}$$

Now let $B_1, \ldots, B_k$ be the solution of the first k equations of system (2.6). The $(k+1)$-st equation is satisfied if D_{k+1} can be chosen such that

$$D_{k+1}(z) = \sum_{l=1}^{k} q_{l=1}^{k} B_l(z). \tag{2.13}$$

From relation (2.12) we know that the only restriction for D_{k+1} is that $D_{k+1}(z)=0(1)$ as $z\to 0$. Hence, relation (2.13) can always be satisfied. Thus, by an appropriate choice of D_{k+1} system (2.6) can always be solved, if $p=k$. □

In general, an accurate choice of J_n, i.e. $J_n=J(x_n, y_n)$, will not increase the order of scheme (1.7)–(2.9). An exception to this rule is made for the one-step scheme. Provided that $J_n=J(x_n, y_n)$, $\varepsilon_2(z)\equiv 0$, and the order of R is greater than one, the order of consistency of the one-step scheme (1.7)–(2.9) is $p=2$. This property may be useful to supply one or more starting values for a multistep scheme (see Verwer [15]). Further we observe that the consistency of R and relation (2.12) imply that $R(z)-1$ and $jD_j(z)-1$, $j=1, \ldots, k-1$, have a zero at $z=0$. Then, according to (2.9) it follows that the rational functions B_l, $l=1, \ldots, k$,

have the same denominator as R. This fact is of practical importance because it means that in scheme (1.7)–(2.9) we have to deal with one matrix inversion.

We derived our formulas through Taylor expansions. After a local linearization argument the same type of formulas may be derived through quadrature formulas (see Lawson [4]), and through interpolation formulas (see Norsett [6]). Moreover, method (1.7)–(2.9) is very closely related to the class of methods proposed by Lambert and Sigurdsson [3]. A more extensive treatment of their class of methods is given by Sigurdsson [10].

3. A-Stability Properties

In the present section we suppose that R is A-acceptable.

Definition 3.1. R is said to be (a) A-acceptable, if $|R(z)|<1$ whenever $\mathrm{Re}(z)<0$; (b) strongly A-acceptable, if it is A-acceptable and satisfies $\lim|R(z)|<1$ as $\mathrm{Re}(z)\to-\infty$; (c) L-acceptable, if it is A-acceptable and satisfies $\lim R(z)=0$ as $\mathrm{Re}(z)\to\infty$.

For further properties of rational approximations to the exponential the reader is referred to Lambert [2].

Consider the $N\times N$ linear test-equation

$$y'=Jy, \tag{3.1}$$

where the eigenvalues of J have negative real part and differ greatly in magnitude, i.e., system (3.1) is stiff. When applied to (3.1), method (1.7) yields the one-step recurrence relation

$$y_{n+1}=R(h_nJ)\,y_n, \tag{3.2}$$

provided that $J_n=J$. This means that the A-stability behaviour of method (1.7), when applied to the general non-linear system (1.1), is governed by R if at each integration step $J_n=J(x_n, y_n)$.

In practice, when non-linear systems are involved, we would not reevaluate J_n at each integration step in order to reduce computational effort. As a consequence, the stability behaviour is only approximately governed by R. To investigate the effect of rough Jacobians, we apply method (1.7)–(2.9) to (3.1) and assume that $J_n=\bar{J}$, an approximation to J. In particular, we assume that the eigenvalues of $\bar{J}$ have negative real part and differ greatly in magnitude. Thus we consider the difference scheme ($h_n=h$)

$$y_{n+1}=R(h\bar{J})\,y_n+h\sum_{l=1}^{k}B_l(h\bar{J})(J-\bar{J})\,y_{n+1-l}. \tag{3.3}$$

We shall derive a bound for the solution of scheme (3.3). In particular, from this bound it will appear that for obtaining favourable stability properties the choice (see (2.7))

$$\varepsilon_j(z)\equiv 0, \qquad j=2,\ldots,k+1, \tag{3.4}$$

is necessary. To begin with we state some preliminary results.

Firstly, a result on the asymptotic behaviour of the functions $B_l(z)$ as $z\to\infty$.

Lemma 3.1. Let R be A-acceptable and suppose condition (3.4) is satisfied. Then the functions B_l defined by (2.9), have a zero at infinity and at least one zero is single-valued.

The proof of the lemma is easily given (see Verwer [16]). On the other hand, it is also easily seen that if condition (3.4) is not satisfied all the B_l have a pole at infinity. For future reference, it follows from relations (2.4) that the functions C_j, $j=0, 1, \ldots$, have no pole at infinity when the conditions of Lemma 3.1 are satisfied.

Secondly, we state a result on the growth of the solution of a class of difference equations.

Lemma 3.2. Let the sequence of vectors $\{y_n\}$ satisfy the difference equation

$$y_{n+1}=(C+hD)\,y_n, \qquad n=0, 1, \ldots, \tag{3.5}$$

where C and D are matrices such that

(a) $\|C^m\|_\infty \leqq Ke^{-\alpha m}$, K and α constants, $K \geqq 1$, m any positive integer,

(b) $\|D\|_\infty = L$,

then, if $\bar{K}=Ke^{\alpha}$,

$$\|y_{n+1}\|_\infty \leqq K\|y_0\|_\infty \exp((n+1)(h\bar{K}L-\alpha)). \tag{3.6}$$

This result is a slight modification of a lemma proposed by Lambert and Sigurdsson [3], which in turn is an extension of Strang's lemma. The lemma can be proved without difficulty by modification of the proof given by Strang [11].

Let $\sigma(C)$ denote the spectral radius of C. We observe that when $\sigma(C)<1$, the constant α occurring in assertion (a) of Lemma 3.2 is positive. As a result, the vector y_n defined by (3.5) tends to zero as $n\to\infty$, if $\alpha > h\bar{K}L$.

Next we write scheme (3.3) in one-step form by introducing the kN-vector $Y_n=[y_n^T, y_{n-1}^T, \ldots, y_{n+1-k}^T]^T$, and the $kN\times kN$ matrices

$$P=\begin{bmatrix} R(h\bar{J}) & 0\cdots 0 & 0 \\ I & 0\cdots 0 & 0 \\ 0 & I\cdots 0 & 0 \\ \vdots & \ddots\ \ddots & \vdots \\ 0 & 0\cdots I & 0 \end{bmatrix} \tag{3.7}$$

and

$$Q=\begin{bmatrix} B_1(h\bar{J})(J-\bar{J}) & \cdots & B_k(h\bar{J})(J-\bar{J}) \\ 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \tag{3.8}$$

where I denotes the $N\times N$ unit matrix. Scheme (3.3) may thus be written as

$$Y_{n+1}=(P+hQ)\,Y_n, \qquad n=k-1, k, \ldots. \tag{3.9}$$

We define $d=\|J-\bar{J}\|_\infty$ and, for any $h>0$, $\beta_l=\|B_l(h\bar{J})\|_\infty$, $l=1, \ldots, k$.

Theorem 3.1. Let R be A-acceptable. Then constants $K \geqq 1$ and $\alpha > 0$ exist such that for $n \geqq k-1$

$$\|Y_{n+1}\|_\infty \leqq K \|Y_{k-1}\|_\infty \exp\left((n+2-k)\,(h\,\overline{K}\,d \sum_{l=1}^{k} \beta_l - \alpha)\right), \tag{3.10}$$

where $K = K e^{\alpha}$.

Proof. The A-acceptability of R implies that $\sigma(P) < 1$. As a result, for the matrix P holds the inequality of assertion (a) of Lemma 3.2, with $\alpha > 0$. The theorem is now easily proved after application of Lemma 3.2 to scheme (3.9). □

Corollary. When

$$h\overline{K}\,d \sum_{l=1}^{k} \beta_l < \alpha, \tag{3.11}$$

the solution y_n of scheme (3.3) tends to zero as $n \to \infty$. Thus, for h small enough, the parasitic solution components will die out for n large. We know that when condition (3.4) is not satisfied all the B_l have a pole at infinity. Because $\sigma(B_l(h\bar{J})) \leqq \beta_l$, it is necessary to satisfy condition (3.4) in order to preclude a considerable reduction in the steplength h as the stiffness increases. According to Lemma 3.1, stiff eigenvalues δ of $\bar{J}$, i.e. $h\,\mathrm{Re}(\delta) \ll -1$, will not enlarge the spectral radii of the matrices $B_l(hJ)$, if condition (3.4) is satisfied. □

In the remainder of this paper we assume that condition (3.4) is satisfied. This implies that the functions B_l are given by

$$\begin{aligned} &\sum_{l=1}^{k} q_{l-1}^{j-1}\, B_l(z) = D_j(z), \qquad j = 1, \ldots, k, \\ &D_1(z) = \frac{R(z)-1}{z}, D_{j+1}(z) = \frac{j D_j(z) - 1}{z}, \qquad j = 1, \ldots, k-1. \end{aligned} \tag{3.12}$$

For scheme (1.7)–(3.12) we may expect a reliable stability behaviour if $J_n \simeq J(x_n, y_n)$. In practice, the stability behaviour of such schemes, when applied to stiff, non-linear systems, turns out to be quite satisfactory (see Verwer [15]).

Stability properties of scheme (1.7)–(3.12) when applied to the linear inhomogeneous equation $y' = Jy + g(x)$, where g is a vector polynomial of degree $k-1$ or less, are discussed by Lawson [4]. Results similar to Lawson's have been given by Norsett [6] for his special class of methods.

Another advantage of condition (3.4) is that the local truncation error (2.3) is minimized (see Verwer [15]). In fact, the choice $\varepsilon_j \equiv 0$, $j = 2, \ldots, k+1$, is identical to the choice (see (2.5))

$$\begin{aligned} &C_j(z) = \frac{1}{j!}, \qquad j = 0, 1, \ldots, k-1, \\ &C_k(z) = \frac{1}{k!} + O(z) \text{ as } z \to 0. \end{aligned} \tag{3.13}$$

4. S-Stability and Stiff-Consistency

Let system (1.1) be stiff. In particular, assume that the eigenvalues of the Jacobian matrices under consideration are located in widely separated clusters. In order to analyse the numerical difficulties that are encountered when solving

such stiff systems, Prothero and Robinson [7] propose the single test-equation

$$y' = g'(x) + \delta(y - g(x)),\ \delta \in \mathbb{C},\ \mathrm{Re}(\delta) \ll 0, \tag{4.1}$$

subject to the initial condition $y = y_0$ at $x = x_0$, where g is any given function with g' bounded. As the solution $y(x)$ of the initial value problem tends to $g(x)$ for $x > x_0$, it is sufficient to analyse the stability and accuracy of numerical approximations to the solution $y \equiv g$ of the stiff equation (4.1). The form of stability analysed, termed S-stability, is an extension of the concept of A-stability, which is related to the test-equation $y' = \delta y$. The concept of stiff-consistency relates the stiffness of equation (4.1) to the accuracy of the integration formula.

In the present section we shall investigate the S-stability and stiff-consistency properties of the adaptive scheme (1.7)–(3.12). For a more extensive discussion about this subject see Prothero and Robinson [7] and Verwer [16]. Let ε_n denote

$$\varepsilon_n = g(x_n) - y_n .$$

Application of scheme (1.7)—(3.12) to equation (4.1) yields an error equation of the form

$$\varepsilon_{n+1} = R(z)\, \varepsilon_n + d_n, \tag{4.2}$$

where $z = h_n\, \delta$, and d_n is defined by

$$d_n = -h_n \left\{ \sum_{l=1}^{k} B_l(z) [g'(x_{n+1-l}) - \delta g(x_{n+1-l})] + (R(z)\, g(x_n) - g(x_{n+1}))/h_n \right\}. \tag{4.3}$$

By putting $\varepsilon_n = 0$ in equation (4.2) we see that d_n represents the local truncation error (see (2.1)) $\mathscr{C}[g(x_n); h_n; \delta]$.

To begin with we give the definition of S-stability which proceeds from Prothero and Robinson: The integration method is said to be S-stable if for any differential equation of the form (4.1) and for any real negative constant δ_0, there exists an $h_0 > 0$ such that $|\varepsilon_{n+1}| < |\varepsilon_n|$, for all stepsizes $0 < h_n < h_0$ and all δ with $\mathrm{Re}(\delta) \leqq \delta_0$.

When $g \equiv c$, c constant, this definition is equivalent to the definition of A-stability.

As observed by the referees the requirement that $|\varepsilon_{n+1}| < |\varepsilon_n|$ is rather unnatural, since $|\varepsilon_n|$ does not tend to zero as $n \to \infty$ (unless the local truncation error does so). In fact, if $|R(z)|$ is close to 1, the requirement that $|\varepsilon_{n+1}| < |\varepsilon_n|$ can be satisfied only for very small values of h_n. Because of this the referees have suggested to modify the definition of S-stability in an appropriate way. In fact, they suggested to modify the inequality to something like

$$|\varepsilon_{n+1}| \leqq \max\{|\varepsilon_n|, |d_n/(1 - R(z))|\}.$$

We now propose the following definition:

Definition 4.1. Let $\bar{h}$ be any positive real number. Then the integration method is said to be S-stable if for any differential equation of the form (4.1) ε_n is uniformly bounded with n for all δ with $\mathrm{Re}(\delta) < 0$ and all $h_n \in (0, \bar{h}]$.

The modification expressed in this definition is similar to the suggestion of the referees and it maintains the essential feature of the concept of S-stability, i.e. the stiff-consistency. In fact, both definitions lead to the same results for our

class of methods. Therefore, we also use the term S-stability. Next we discuss the concept of stiff-consistency. When $\mathrm{Re}(\delta) \to -\infty$, the solution $y(x) \to g(x)$ for any fixed $x > x_0$, regardless of the initial condition at $x = x_0$. Thus for stiff eigenvalues, i.e. $\mathrm{Re}(z) \ll 0$, a good measure of the local accuracy of the method, when applied to problem (4.1), is provided by $d_n = \mathscr{C}[g(x_n); h_n; \delta]$. According to Prothero and Robinson [7], we can derive an asymptotic relation for d_n in the limit: $\mathrm{Re}(z) \to -\infty$ and $h_n \to 0$, which is of the form

$$d_n \sim K_0 h_n^{p_h+1} g^{(p_h+1)}(x_n) + \sum_{j=1}^{s} K_j z^{t_j} h_n^{p_j+1} g^{p_j+1}(x_n), \tag{4.4}$$

where K_j, $j=0, \ldots s$ are real constants not equal to zero; p_h, p_j, $\mathrm{j}=1, \ldots, s$ are non-negative integers; s is a positive integer; and t_j, $j=1, \ldots, s$ are integers. Using the definition of Prothero and Robinson, the method should be stiffly accurate when in relation (4.4) the integers t_j are negative and the constant K_0 is equal to zero. However K_0 is not equal to zero, indicating that the definition of stiff-accuracy can not be applied to our class of methods. Therefore we proceed with the following definition:

Definition 4.2. Let $\bar{h}$ be any positive real number, and let $z = h_n\delta$. Then the integration method is said to be stiffly consistent if d_n/h_n is uniformly bounded on $\{(h_n, z) \mid h_n \in (0, \bar{h}], \mathrm{Re}(z) < 0\}$.

The term stiff-consistency, instead of stiff-accuracy, has been suggested by Prothero [8].

The S-stability of the integration method is governed by the error equation (4.2). To establish conditions for S-stability and stiff-consistency we need some preliminary results.

Lemma 4.1. Let $\alpha(z)$, $\beta_n(z)$, $n=0, 1, \ldots$, be complex valued functions defined on some region S in $\mathbb{C}$. Define for all complex ε_0 the recurrence relation

$$\varepsilon_{n+1} = \alpha(z)\, \varepsilon_n + \beta_n(z), \qquad n=0, 1, \ldots .$$

Then, $\varepsilon_n(z)$ is uniformly bounded in n and $z \in S$, if

(a) $|\alpha(z)| < 1$ on S, and,

(b) $\beta_n(z)/(1 - |\alpha(z)|)$ is uniformly bounded in n and $z \in S$.

The proof of this lemma is trivial.

Lemma 4.2. Let R be A-acceptable. Then for any fixed $\bar{h} > 0$, d_n/h_n is uniformly bounded on $(0, \bar{h}] \times \{z \mid \mathrm{Re}(z) < 0\}$.

Proof. Because the functions $zB_l(z)$, $l=1, \ldots, k$, and $R(z)$ are uniformly bounded on $\{z \mid \mathrm{Re}(z) < 0\}$, we only have to consider d_n/h_n as $h_n \to 0$. It follows from expansion (2.3) and relations (3.13) that for any z, $\mathrm{Re}(z) < 0$, d_n may be formally expanded as

$$d_n = \sum_{j=k}^{\infty} [C_j(z) - 1/j!]\, h_n^j\, g^{(j)}(x_n) \text{ as } h_n \to 0, \tag{4.5}$$

where $k \geqq 1$. The proof is now easily completed by observing that the functions $C_j(z)$, $j=k, k+1, \ldots$, are also uniformly bounded on $\{z \mid \mathrm{Re}(z) < 0\}$. □

The conditions for S-stability and stiff-consistency are expressed by the following theorem, which is readily proved with the help of the lemmas given above.

Theorem 4.1 The adaptive scheme (1.7)–(3.12) is

(a) stiffly consistent if R is A-acceptable,

(b) S-stable if R is strongly A-acceptable.

When using the original definition of S-stability one can prove that the strong A-acceptability of R is also necessary (see Verwer [16]).

Suppose C_k may be expanded as $C_k(z)=c_k^{(0)}+c_k^{(1)}\,z^{-1}+\cdots$, as $\mathrm{Re}(z)\to-\infty$, with $c_k^{(0)}\neq 1/k!$, $c_k^{(1)}\neq 0$. Then the asymptotic relation for d_n reads (cf. (4.4))

$$d_n\sim[c_k^{(0)}-1/k!+c_k^{(1)}\,z^{-1}]\;h_n^k\,g^{(k)}(x_n),\ h_n\to 0 \text{ and } \mathrm{Re}(z)\to-\infty. \tag{4.6}$$

It follows from relation (4.6) that scheme (1.7)–(3.12), when A-acceptable, exhibits a good stiff-consistency behaviour.

Observe that when R is L-acceptable and $\mathrm{Re}(z)\ll -1$, the local truncation errors d_j, $j<n$, are very quickly damped out. In such a situation the global errors ε_{n+1} approximately equal d_n.

In order to obtain stiff-consistency and S-stability the uniform boundedness of the functions B_l on $\{z\,|\,\mathrm{Re}(z)<0\}$ has shown to be necessary, implying the necessity of condition (3.4).

5. Numerical Examples

In order to give an indication of the stability- and accuracy properties of methods from the class (1.7)–(3.12), we apply a third order formula from this class to three stiff, non-linear problems which are well-known in the literature.

As an example, we choose for R the third order, L-acceptable Padé-approximation

$$R(z)=\frac{1+\frac{1}{3}z}{1-\frac{2}{3}z+\frac{1}{6}z^2}. \tag{5.1}$$

We shall integrate with constant steplength. Then B_1, B_2 and B_3 are given by

$$B_1(z)=\frac{\frac{23}{12}-\frac{1}{2}z}{1-\frac{2}{3}z+\frac{1}{6}z^2},\qquad B_2(z)=\frac{-\frac{4}{3}+\frac{1}{2}z}{1-\frac{2}{3}z+\frac{1}{6}z^2},$$
$$B_3(z)=\frac{\frac{5}{12}-\frac{1}{6}z}{1-\frac{2}{3}z+\frac{1}{6}z^2}. \tag{5.2}$$

The two additional starting values for the three-step scheme are computed by a second order starting mechanism, which makes use of the exact Jacobian at $x=x_0$.

This starting mechanism is used in an ALGOL-60 integration procedure, of which the integration formula is generated by (5.1) and belongs to the class

(1.7)–(3.12). This procedure performs a stepsize control based on the non-linearity of the differential equation and has been applied with succes to a number of stiff non-linear systems. It is also provided with a mechanism to control the reevaluating of the Jacobian matrix. Computational results of this procedure are given in Verwer [16].

Here we consider the following problems (see also Lambert and Sigurdsson [3]).

I. (Liniger and Willoughby [5]).

$$y_1' = 10\,y_2 - (60 - 0.125\,x)\,y_1 + 0.125\,x,$$
$$y_2' = 0.2\,(y_1 - y_2),$$
$$y_1(0) = y_2(0) = 0,\ 0 \leqq x \leqq 400,$$
reference solution at $x = 400$:
$$\tilde{y}_1 = 0.27110701_{10}2,\ \tilde{y}_2 = 0.22242211_{10}2.$$

II. (Liniger and Willoughby [5]).

$$y_1' = 0.01 - [1 + (y_1 + 1000)\,(y_1 + 1)]\,(0.01 + y_1 + y_2),$$
$$y_2' = 0.01 - (1 + y_2^2)\,(0.01 + y_1 + y_2),$$
$$y_1(0) = y_2(0) = 0,\ 0 \leqq x \leqq 100,$$
reference solution at $x = 100$:
$$\tilde{y}_1 = -0.99164207,\ \tilde{y}_2 = 0.98333636.$$

III. (Gear [1]).

$$y_1' = -0.013\,y_2 - 1000\,y_1\,y_2 - 2500\,y_1\,y_3,$$
$$y_2' = -0.013\,y_2 - 1000\,y_1\,y_2,$$
$$y_3' = -2500\,y_1\,y_3,$$
$$y_1(0) = 0,\ y_2(0) = y_3(0) = 1,\ 0 \leqq x \leqq 50,$$
reference solution at $x = 50$:
$$\tilde{y}_1 = -0.189_{10} - 5,\ \tilde{y}_2 = 0.59765470,\ \tilde{y}_3 = 0.14023434_{10}1.$$

Let n denote the number of steps necessary to integrate the given interval. For problem I $n = 400$, for problem II $n = 1000$ and for problem III $n = 500$. In order to illustrate the advantage of consistency conditions allowing inaccurate Jacobian matrices, the three problems are solved for several values of m, where m denotes the number of steps per Jacobian evaluation. Because the integration process is always started with an evaluation of the Jacobian, $m = n$ means that the integration has been performed with one evaluation at $x = x_0$.

In Table 1 we give for each j-th component the number of significant digits $sd_j = -{}^{10}\log$ (*absolute error*) at the endpoint of the given interval. The letter u

Table 1. Table of results

	m	1	5	10	25	n
I	sd_1	2.3	2.3	2.2	2.1	1.0
	sd_2	2.7	2.6	2.6	2.4	1.3
II	sd_1	5.2	5.3	5.3	5.3	u
	sd_2	5.2	5.2	5.2	5.3	u
III	sd_1	8.5	8.5	8.5	8.5	8.5
	sd_2	7.3	6.8	6.8	6.8	6.8
	sd_3	7.3	6.8	6.8	6.8	6.8

refers to an unstable result. The calculations have been performed on a Cyber 73–28 computer.

The results of these computations indicate that methods from the class (1.7)–(3.12) are useful for solving stiff, non-linear systems over a large range of integration. In general we obtain stable solutions with a relatively few number of Jacobian evaluations. In practice it is important to reduce this number as much as possible because each reevaluation involves a LU-decomposition.

It is of interest to observe that for $m=1$ our results may be compared with the results, given by Lambert and Sigurdsson [3], of two third order methods which are closely related to our third order method. For $m \neq 1$ no results are given in their paper.

Acknowledgements. The author wishes to thank P. J. van der Houwen for suggesting this research project and for many useful comments during the course of the investigation

References

1. Gear, C. W.: Numerical integration of stiff ordinary differential equations. Report 221, Dept. of Computer Science, Univ. of Illinois, Urbana (1967)
2. Lambert, J. D.: Computational methods in ordinary differential equations. London: Wiley 1973
3. Lambert, J. D., Sigurdsson, S. T.: Multistep methods with variable matrix coefficients. SIAM J. Numer. Anal. **9**, 715–733 (1972)
4. Lawson, J. D.: Some numerical methods for stiff ordinary and partial differential equations, Proc. Second Manitoba Conf. on Numerical Math. (R. Thomas, H. Williams, eds.), Univ. of Manitoba, Winnipeg (1972)
5. Liniger, W., Willoughby, R. A.: Efficient numerical integration of stiff systems of ordinary differential equations. Tech. Report RC-1970. IBM Thomas J. Watson Research Centre, Yorktown Heights, New York (1967)
6. Nørsett, S. P.: An A-stable modification of the Adams-Bashforth methods. Lecture Notes in Math. 109 (A. Dold, B. Echmann, eds.). Berlin-Heidelberg-New York: Springer 1969
7. Prothero, A., Robinson, A.: On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. Math. Comp. **28**, 145–162 (1974)
8. Prothero, A.: Private communication
9. Rosenbrock, H. H.: Some general implicit processes for the numerical solution of differential equations. Computer J. **5**, 329–330 (1963)
10. Sigurdsson, S. T.: Multistep methods with variable matrix coefficients for systems of ordinary differential equations. Report 1973-04, Dept. Comp. Sc., Chalmers Inst. of Techn. and the Univ. of Gøteborg, Sweden (1973)
11. Strang, G.: Accurate partial difference methods. II. Nonlinear problems. Numer. Math. **6**, 37–46 (1964)
12. Van der Houwen, P. J.: A note on two-step integration formulas of third order accuracy with prescribed stability function. Report NR 26/72, Mathematisch Centrum, Amsterdam (1972)
13. Van der Houwen, P. J.: Construction of integration formulas for initial value problems. Amsterdam: North-Holland 1976

14. Van der Houwen, P. J., Verwer, J. G.: Generalized linear multistep methods. I. Development of algorithms with zero-parasitic roots. Report NW 10/74, Mathematisch Centrum, Amsterdam (1974)
15. Verwer, J. G.: Generalized linear multistep methods. II. Numerical applications. Report NW 12/74, Mathematisch Centrum, Amsterdam (1974)
16. Verwer, J. G.: S-stability and stiff-accuracy for two classes of generalized integration methods. Report NW 15/75, Mathematisch Centrum, Amsterdam (1975)

J. G. Verwer
Mathematisch Centrum
2de Boerhaavestraat 49
Amsterdam (Oost), The Netherlands

Numer. Math. 27, 359—370 (1977)

S-Stability Properties for Generalized Runge-Kutta Methods

J. G. Verwer

Received October 10, 1975

Summary. A class of generalized Runge-Kutta methods is considered for the numerical integration of stiff systems of ordinary differential equations. These methods are characterized by the fact that the coefficients of the integration formulas are matrices depending on the Jacobian, or on an approximation to the Jacobian. Special attention is paid to stability aspects. In particular, the *S*-stability properties of the method are investigated. The concept of internal stability is discussed. Internal stability imposes conditions on intermediate results in the Runge-Kutta scheme. Some numerical examples are discussed.

1. Introduction

Let

$$y' = f(x, y), \qquad y(x_0) = y_0, \tag{1.1}$$

represent an initial value problem for a stiff system of ordinary differential equations of which the real vector function $f(x,y)$ is sufficiently differentiable. Most integration methods for stiff systems are implicit and demand the use of some Newton type iteration. As a consequence, the Jacobian matrix $\partial f/\partial y$ has to be evaluated. It is therefore natural to investigate integration schemes which incorporate the Jacobian directly.

An example of such a scheme is the generalized Runge-Kutta scheme, i.e., a Runge-Kutta scheme where the Jacobian matrix is introduced directly into the coefficients of the integration formula. The idea to introduce the Jacobian directly into the coefficients of the Runge-Kutta formula proceeds from Rosenbrock [5]. He proposed a special class of generalized Runge-Kutta methods, which in principle require one evaluation of the Jacobian matrix at each Runge-Kutta stage. From a practical point of view such formulas are unattractive, because, in general, each evaluation of the Jacobian matrix also involves an LU-decomposition.

In this paper we consider formulas of a type first proposed by Van der Houwen [6]. These formulas require at most one evaluation of the Jacobian matrix per integration step. In their general form, they may be obtained from ordinary explicit Runge-Kutta formulas by replacing the scalar coefficients by rational functions of the Jacobian matrix. For a treatment of existing generalized methods, as well as for references to related papers, the reader is referred to Van der Houwen [7].

Due to the presence of the rational functions, we are able to develop formulas with stability properties that are needed when integrating stiff systems. The requirement of stability is a basic difficulty in the numerical solution of stiff systems.

Until now, the stability analysis of our class of methods, as well as others, has centred largely on the stability of numerical solutions to the scalar test-model (see e.g. Dahlquist [1])

$$y' = \delta y, \qquad \delta \in \mathbb{C}, \qquad \mathrm{Re}(\delta) < 0. \tag{1.2}$$

However, it is well known that test-model (1.2) is rather limited for the stability analysis of integration methods for stiff and strongly non-linear systems. For the analysis of implicit Runge-Kutta methods, Prothero and Robinson [4] recently proposed the scalar test-model

$$y' = g'(x) + \delta(y - g(x)), \qquad \delta \in \mathbb{C}, \qquad \mathrm{Re}(\delta) < 0, \tag{1.3}$$

where g is any given, sufficiently differentiable function. It turned out, that for implicit Runge-Kutta methods, this test-model is more appropriate to predict the behaviour of numerical solutions to stiff, non-linear systems. In their paper, Prothero and Robinson analyze the stability of numerical approximations to the solution $y \equiv g$ of (1.3), and define the so-called concept of S-stability. They also analyze the accuracy of approximations to g, especially for stiff eigenvalues, and define the concept of stiff accuracy and stiff order.

In this paper we shall discuss the S-stability properties for our generalized Runge-Kutta method. Following the approach of Prothero and Robinson, we also define the concept of stiff consistency, which is related to the concept of stiff accuracy mentioned above. We shall however not adopt the ideas of Prothero and Robinson concerning the stiff order, as this has shown to be of no practical relevance for our class of methods.

Further, we discuss the concept of internal stability. The usual approach is to require stability only for the numerical solution at step points, while internal stability imposes also conditions on intermediate results. The idea is of importance for Runge-Kutta schemes of which the solution is built up successively. We shall require internal stability for test-model (1.3), thus deriving the property of internal S-stability. An immediate consequence of this property is that at each stage of the Runge-Kutta scheme, stiff components in the numerical solution are suppressed.

To illustrate the relevance of the various stability concepts, we shall present some numerical results in the last section of this paper. These results clearly indicate that S-stable, generalized Runge-Kutta methods are superior to A-stable ones.

2. Generalized Runge-Kutta Methods

Let h_n denote the steplength, i.e., $h_n = x_{n+1} - x_n$. Let y_n denote the numerical approximation to the analytical solution $y(x_n)$ of (1.1) at $x = x_n$, and let J_n denote the Jacobian matrix $\partial f/\partial y$, evaluated at (x_n, y_n). Further define

$$A_{j,l}, \; j = 1, \ldots, m; \quad l = 0, \ldots, j-1, \tag{2.1}$$

to be rational functions with real coefficients. For the purpose of our analysis it is assumed that the rational functions have their poles in the right half-plane.

Our generalized, m-point Runge-Kutta method is now defined by

$$y_{n+1}=y_n+h_n\phi(x_n,y_n,h_n,J_n)=y_n+\sum_{j=0}^{m-1}\Lambda_{m,j}(h_nJ_n)\,k_n^{(j)},$$
$$k_n^{(j)}=h_nf(x_n+\mu_jh_n,y_n+\sum_{l=0}^{j-1}\Lambda_{j,l}(h_nJ_n)\,k_n^{(l)}), \tag{2.2}$$

where the step parameters μ_j are given by

$$\mu_j=\sum_{l=0}^{j-1}\Lambda_{j,l}(0),\qquad j=0,\ldots,m-1. \tag{2.3}$$

We shall define the consistency of method (2.2) in the usual way.

Definition 2.1. Method (2.2) is said to be consistent of order p, at $x=x_n$, if p is the largest integer such that

$$y(x_{n+1})-y(x_n)-h_n\phi(x_n,y(x_n),h_n,J_n)=O(h_n^{p+1}),\quad h_n\to 0, \tag{2.4}$$

where $y(x)$ is the analytical solution of (1.1).

Moreover, the method is said to be consistent if $p\geqq 1$. It is clear that our method is consistent when $\mu_m=1$, where μ_m is defined according to (2.3), i.e.,

$$\mu_m=\sum_{l=0}^{m-1}\Lambda_{m,l}(0). \tag{2.5}$$

The consistency conditions for order $p>1$ may be derived though Taylor expansions about x_n. In this paper we simply assume that we are dealing with consistent methods.

Next we apply method (2.2) to test-model (1.2). This yields the one-step recurrence

$$y_{n+1}=R(z)\,y_n,\quad z=h_n\delta, \tag{2.6}$$

where R is a rational function representing the stability function of the one-step method.

Definition 2.2. R is said to be (a) A-acceptable, if $|R(z)|<1$ whenever $\mathrm{Re}(z)<0$; (b) strongly A-acceptable, if it is A-acceptable and satisfies $\lim|R(z)|<1$ as $\mathrm{Re}(z)\to-\infty$; (c) L-acceptable, if it is A-acceptable and satisfies $\lim R(z)=0$ as $\mathrm{Re}(z)\to-\infty$.

For further properties of rational approximations to the exponential, the reader is referred to Lambert [3].

Next we introduce the matrix function $\Lambda\colon\mathbb{C}\to L(\mathbb{C}^m)$, defined by

$$\Lambda\equiv\begin{bmatrix}0 & 0 & . & . & . & 0\\ \Lambda_{1,0} & 0 & . & . & . & 0\\ \Lambda_{2,0} & \Lambda_{2,1} & & & & .\\ . & & . & & & .\\ . & & & . & & .\\ . & & & & . & .\\ \Lambda_{m-1,0} & \Lambda_{m-1,1} & & \Lambda_{m-1,\,m-2} & & 0\end{bmatrix}, \tag{2.7}$$

and the vector function Λ_m: $\mathbb{C} \to \mathbb{C}^m$, defined by

$$\Lambda_m \equiv [\Lambda_{m,0}, \Lambda_{m,1}, \ldots, \Lambda_{m,m-1}]^T. \tag{2.8}$$

The matrix $\Lambda(z)$ and the vector $\Lambda_m(z)$ shall be used in the stability analysis of the method. Moreover, the method may be formally characterized by the $(m+1)*m$ matrix

$$\left[\frac{\Lambda(z)}{\Lambda_m^T(z)}\right]. \tag{2.9}$$

In the remainder of this paper we also need a lemma about the inverse of the matrix $I - z\Lambda(z)$. The proof of this lemma may be obtained by elementary matrix algebra.

Lemma 2.1. Each element of the inverse of the matrix $I - z\Lambda(z)$ is uniformly bounded on $\{z \in \mathbb{C} \mid \mathrm{Re}(z) < 0\}$, if each element of the matrix $\Lambda(z)$ has a zero at infinity.

Before proceeding to the next section we make two remarks about the practical application of the method. Firstly, the number of LU-decompositions to make depends on the number of different denominators of the rational functions. In practice therefore, it is in general advisable to use methods where the rational functions share the same denominator. Secondly, method (2.2) can also be used with J_n replaced by an approximation to the Jacobian. In practice this is important in order to reduce the number of evaluations of the Jacobian. When using approximations, one needs consistency conditions which are independent of J_n. Moreover, one should be aware of the fact that, when using approximations, the stability function of the method is only approximately given by R (see (2.6)).

3. S-Stability and Stiff Consistency

In order to analyze the numerical difficulties that are encountered when solving stiff and rather non-linear problems with implicit Runge-Kutta methods, Prothero and Robinson [4] propose the scalar test-model (1.3), i.e.,

$$y' = g'(x) + \delta(y - g(x)). \tag{3.1}$$

It turns out that this test-model is also of importance for our generalized Runge-Kutta methods. For a justification of (3.1), we refer to the paper of Prothero and Robinson, or to Verwer [9]. Here we confine ourselves to a brief discussion of the definitions of S-stability and stiff consistency.

Subject to the initial condition $y = y_0$ at $x = x_0$, (3.1) has the solution

$$y(x) = e^{\delta(x-x_0)}[y(x_0) - g(x_0)] + g(x), \tag{3.2}$$

which tends to the solution g as $x \to \infty$. This means that it is sufficient to require stability of numerical approximations to (3.2) with respect to solution g.

Let us assume that the integration method is applied to an equation of the form (3.1). Define

$$\varepsilon_n = g(x_n) - y_n.$$

Then, according to Prothero and Robinson [4], the integration method is said to be *S*-stable if for any real negative constant δ_0, there exists an $h_0>0$ such that $|\varepsilon_{n+1}|<|\varepsilon_n|$, for all $0<h_n<h_0$ and all δ with $\mathrm{Re}(\delta)\leqq\delta_0$.

The requirement, in the above definition, that $|\varepsilon_{n+1}|<|\varepsilon_n|$ is rather unnatural since $|\varepsilon_n|$ does not tend to zero as $n\to\infty$ (unless the local truncation error does so). Because of this, we prefer a definition which is given in Verwer [11]. This definition reads:

Definition 3.1. Let $\bar{h}$ by any positive real number. Then the integration method is said to be *S*-stable, if for a differential equation of the form (3.1) ε_n is uniformly bounded with n for all δ with $\mathrm{Re}(\delta)<0$ and all $h_n\in(0,\bar{h})$.

The modification expressed in this definition maintains the essential feature of the concept of *S*-stability, i.e. the stiff-consistency. In fact, both definitions lead to the same results for our class of methods (see Verwer [9]). Therefore we also use the term *S*-stability.

It is readily seen that for g constant Definition 3.1 is equivalent to the definition of *A*-stability. If the conditions only hold for $|\arg(-h\delta)|<\alpha$, the method is said to be $S(\alpha)$-stable.

An important role in the *S*-stability theory is played by the truncation error (compare (2.4))

$$d_n=g(x_{n+1})-g(x_n)-h_n\phi(x_n,g(x_n),h_n,\delta). \tag{3.3}$$

As $\mathrm{Re}(\delta)\to-\infty$, solution (3.2) tends to $g(x)$ for any fixed $x>x_0$. Thus, for stiff eigenvalues d_n provides a very good estimate for the truncation error with respect to solution (3.2). In order to distinguish from the local truncation error (2.4), expression (3.3) might therefore be called the asymptotic truncation error.

Prothero and Robinson now suggest to define a concept of stiff order by means of the asymptotic relation for d_n, in the limit: $h_n\to0$, $\mathrm{Re}(h_n\delta)\to-\infty$. The reason for this is to design integration formulas which give more accurate results on stiff, non linear problems. For it is well known that for these problems the local truncation error in the usual sense behaves rather badly (compare Van Veldhuizen [8]). Unfortunately, for our class of methods the concept of stiff order has shown to be of no great practical relevance (see Verwer [9, 10]). Therefore, we shall confine ourselves to the concept of stiff consistency.

Definition 3.2. Let $\bar{h}$ be any positive real number, and let $z=h_n\delta$. Then method (2.2) is said to be stiffly consistent if d_n/h_n is uniformly bounded on

$$\{(h_n,z)\,|\,h_n\in(0,\bar{h}],\quad \mathrm{Re}(z)<0\}.$$

Before applying method (2.2) to Equation (3.1) we introduce the notation

$$g_{n,j}=g(x_n+\mu_j h_n),\ g'_{n,j}=g'(x_n+\mu_j h_n)\quad j=0,\ldots,m-1, \tag{3.4}$$

and the m-vectors

$$g_n=[g_{n,0},\ldots,g_{n,m-1}]^T,\quad g'_n=[g'_{n,0},\ldots,g'_{n(m-1}]^T,\quad e=[1,\ldots,1]^T. \tag{3.5}$$

Further, for scalar equations we introduce the m-vector

$$k_n=[k_n^{(0)},\ldots,k_n^{(m-1)}]^T. \tag{3.6}$$

For Equation (3.1) the increment functions $k_n^{(j)}$ read

$$k_n^{(j)} = h_n g'_{n,j} - z\,(g_{n,j} - y_n - \sum_{l=0}^{j-1} \Lambda_{j,l}(z)\, k_n^{(l)}), \tag{3.7}$$

where $z = h_n \delta$. Now it is readily seen that the increment vector k_n can be expressed as

$$k_n = (I - z\Lambda(z))^{-1}\,[z y_n e + h_n g'_n - z g_n]. \tag{3.8}$$

Thus, for y_{n+1} we find the expression

$$y_{n+1} = [1 + z\Lambda_m(z)\,(I - z\Lambda(z))^{-1} e] y_n + \Lambda_m(z)\,(I - z\Lambda(z))^{-1}(h_n g'_n - z g_n). \tag{3.9}$$

As a result, the stability function of method (2.2) is given by

$$R(z) = 1 + z\Lambda_m(z)\,(I - z\Lambda(z))^{-1} e. \tag{3.10}$$

An explicit expression for R can be found in Verwer [9]. After a short calculation, the difference equation for the error ε_n is now found to be

$$\varepsilon_{n+1} = R(z)\,\varepsilon_n + d_n, \tag{3.11}$$

where the truncation error d_n, as defined by expression (3.3), reads

$$d_n = g(x_{n+1}) - R(z)\,g(x_n) - \Lambda_m(z)\,(I - z\Lambda(z))^{-1}(h_n g'_n - z g_n). \tag{3.12}$$

In order to establish conditions for S-stability we need the following lemma.

Lemma 3.1. Let $\alpha(z)$, $\beta_n(z)$, $n = 0, 1, \ldots$, be complex valued functions defined on some region S in $\mathbb{C}$. Define for all complex ε_0 the recurrence relation

$$\varepsilon_{n+1} = \alpha(z)\,\varepsilon_n + \beta_n(z), \qquad n = 0, 1, \ldots.$$

Then, $\varepsilon_n(z)$ is uniformly bounded in n and $z \in S$, if

(a) $|\alpha(z)| < 1$ on S, and,

(b) $\beta_n(z) \,/\, (1 - |\alpha(z)|)$ is uniformly bounded in n and $z \in S$.

The proof of this lemma is trivial. By applying this lemma to the error equation (3.11) we immediately arrive at

Theorem 3.1. The generalized Runge-Kutta method (2.2) is S-stable if it is strongly A-stable and stiffly consistent.

Here the important role of the asymptotic truncation error becomes clear. When admitting increasing stiffness, i.e., $\mathrm{Re}(\delta) \to -\infty$, the asymptotic error d_n is not necessarily bounded for a strongly A-stable method. This means that such a method behaves badly when applied to strongly non-linear problems. When using the definition of Prothero and Robinson, one can prove that strong A-stability and stiff consistency is necessary for S-stability (see Verwer [9]).

Let us establish stiff consistency conditions.

Theorem 3.2. The generalized Runge-Kutta method (2.2) is stiffly consistent if the coefficient functions $\Lambda_{j,l}$, $j = 1, \ldots, m$; $l = 0, \ldots, j-1$, have a zero at infinity.

Proof. Using (3.10), expression (3.12) can be written as

$$d_n = g(x_{n+1}) - g(x_n) - A_m(z) \\ (I - zA(z))^{-1} h_n g'_n - zA_m(z)\,(I - zA(z))^{-1}(g(x_n)e - g_n). \tag{3.13}$$

The mean value theorem, applied to the components $g_{n,j}$ of g_n and to g, yields

$$g_{n,j} = g(x_n) + \mu_j h_n g'(x_n + \theta_j \mu_j h_n), \qquad 0 < \theta_j < 1, \tag{3.14}$$

and

$$g(x_{n+1}) = g(x_n) + h_n g'(x_n + \theta_m h_n), \qquad 0 < \theta_m < 1, \tag{3.15}$$

respectively. Thus, d_n can be written as

$$d_n = h_n \{ g'(x_n + \theta_m h_n) - A_m(z)\,(I - zA(z))^{-1}(g'_n - z\bar{g}'_n) \}, \tag{3.16}$$

where

$$\bar{g}'_n = [\mu_0 g'(x_n + \theta_0 \mu_0 h_n), \ldots, \mu_{m-1} g'(x_n + \theta_{m-1}\mu_{m-1} h_n)]^T. \tag{3.17}$$

From Lemma 2.1 it now follows that for any positive $\bar{h}$, d_n/h_n is uniformly bounded on $\{(h_n, z) \mid h_n \in (0, \bar{h}],\ \mathrm{Re}(z) < 0\}$, establishing stiff consistency. □

By means of Theorems 3.1–3.2, we finally have

Theorem 3.3. The generalized Runge-Kutta method (2.2) is S-stable, if

(a) it is strongly A-stable,

(b) its coefficient functions $A_{j,l}$, $j = 1, \ldots, m$; $l = 0, \ldots, j-1$, have a zero at infinity.

4. Internal S-Stability

In the preceding section we discussed the S-stability properties of our class of methods. The concept of S-stability is directly relevant to the numerical solution of stiff, non-linear systems. In this section we discuss the concept of internal S-stability. Internal S-stability is a stronger form of numerical stability than S-stability.

We begin by defining the vectors $y^{(j)}_{n+1}$, $j = 1, \ldots, m-1$, by means of relation

$$k^{(j)}_n = h_n f(x_n + \mu_j h_n,\ y^{(j)}_{n+1}), \qquad j = 1, \ldots, m, \tag{4.1}$$

and we rewrite scheme (2.2) in the form

$$\begin{aligned} y^{(0)}_{n+1} &= y_n, \\ y^{(j)}_{n+1} &= y_n + h_n \sum_{l=0}^{j-1} A_{j,l}(h_n J_n) f(x_n + \mu_l h_n,\ y^{(l)}_{n+1}), \quad j = 1, \ldots, m, \\ y_{n+1} &= y^{(m)}_{n+1}. \end{aligned} \tag{4.2}$$

In order to discuss the internal stability, representation (4.2) is more convenient than representation (2.2).

The usual approach in the stability analysis of methods for stiff problems, is to relate the stiffness to the numerical solution only at the step points x_n. For Runge-Kutta schemes of which the solution is built up successively, and which are applied to strongly non-linear systems, it is also of interest to relate the stiffness to the intermediate approximations $y^{(j)}_{n+1}$. When doing this, we take account of the fact that generally the Jacobian varies over the step interval.

We shall define internal stability properties for the S-stability test-model. Of course, it is more realistic to consider test-equations in which a variable Jacobian occurs. However, for integration schemes of the Runge-Kutta type it is almost impracticable to do this.

Definition 4.1. The integration method is said to be internally S-stable, if at each j-th stage, $j=1, \ldots, m$, the corresponding scheme of stage j is S-stable.

If the above conditions only hold for $|\arg(-h\delta)|<\alpha$, then the method is said to be internally $S(\alpha)$-stable. We can also define internal stability properties for the A-stability test-model (1.2). For g constant, Definition 4.1 is then equivalent to the definition of internal A-stability, which means that at each stage of the scheme stiff components are suppressed.

According to Definition 4.1, internal S-stability means that at each stage we calculate an S-stable approximation $y_{n+1}^{(j)}$. As a result, conditions for internal S-stability can be derived immediately by applying Theorem 3.3 to each stage:

Theorem 4.1. Let $R^{(j)}$ represent the stability function for the first j stages, $j=1, \ldots, m$. Then the generalized Runge-Kutta scheme (2.2) is internally S-stable, if

(a) $R^{(j)}$, $j=1, \ldots, m$, is strongly A-acceptable,

(b) the coefficient functions $A_{j,l}$, $j=1, \ldots, m$; $l=0, \ldots, j-1$, have a zero at infinity.

The internal stability functions $R^{(j)}$ may be obtained from relation (3.10) after adjusting the order of the characterizing matrix. The relevance of the concept of internal S-stability is shown in the next section by means of some numerical examples.

5. Numerical Examples

In this section we present some numerical results with the aim of showing that S-stable methods from the generalized Runge-Kutta class are generally superior to the A-stable ones. We consider three second order schemes which all may be looked upon as generalized versions of the explicit trapezoidal method

$$y_{n+1}=y_n+\tfrac{1}{2}h_n f(x_n,y_n)+\tfrac{1}{2}h_n f(x_n+h_n, y_n+h_n f(x_n,y_n)). \tag{5.1}$$

In order to describe the methods, we use the characterizing matrix (2.9). The methods read:

I.

$$\begin{bmatrix} 0 & 0 & 0 \\ \dfrac{1+(2\sqrt{2}-3)z}{(1+(\frac{1}{2}\sqrt{2}-1)z)^2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}, \tag{5.2}$$

$$R^{(1)}(z)=\frac{1+(\sqrt{2}-1)z+(\sqrt{2}-\frac{3}{2})z^2}{(1+(\frac{1}{2}\sqrt{2}-1)z)^2}, \tag{5.3}$$

$$R^{(2)}(z)=\frac{1+(\sqrt{2}-1)z}{(1+(\frac{1}{2}\sqrt{2}-1)z)^2}. \tag{5.4}$$

The stability function $R^{(2)}$ is L-acceptable, while the internal stability function $R^{(1)}$ is $A\left(\frac{2\pi}{5}\right)$-acceptable. The method is not S-stable, as it is not stiffly consistent. The lack of stiff consistency is due to the fact that the coefficient functions $A_{2,0}$ and $A_{2,1}$ have no zero at infinity.

II.

$$\begin{bmatrix} 0 & 0 & 0 \\ \dfrac{1}{1+(\frac{1}{2}\sqrt{2}-1)z} & 0 & 0 \\ \dfrac{\frac{1}{2}+(\frac{3}{4}\sqrt{2}-\frac{3}{2})z}{(1+(\frac{1}{2}\sqrt{2}-1)z)^2} & \dfrac{\frac{1}{2}}{1+(\frac{1}{2}\sqrt{2}-1)z} & 0 \end{bmatrix}, \tag{5.5}$$

$$R^{(1)}(z)=\frac{1+\frac{1}{2}\sqrt{2}z}{1+(\frac{1}{2}\sqrt{2}-1)z}, \tag{5.6}$$

$$R^{(2)}(z)=\frac{1+(\sqrt{2}-1)z}{(1+(\frac{1}{2}\sqrt{2}-1)z)^2}. \tag{5.7}$$

The stability function $R^{(2)}$ is L-acceptable; however, the internal stability function $R^{(1)}$ is not even $A(0)$-acceptable, as $R^{(1)}(z)\to\approx -2.4$ if $\mathrm{Re}(z)\to-\infty$. Thus, because of the fact that all the coefficient functions have a zero at infinity, the method is S-stable, but not internally S-stable. Note that method I and II have the same stability function.

III.

$$\begin{bmatrix} 0 & 0 & 0 \\ \dfrac{1}{1-z} & 0 & 0 \\ \dfrac{\frac{1}{2}-z}{(1-z)^2} & \dfrac{\frac{1}{2}-z}{(1-z)^2} & 0 \end{bmatrix}, \tag{5.8}$$

$$R^{(1)}(z)=\frac{1}{1-z}, \tag{5.9}$$

$$R^{(2)}(z)=\frac{1-2z+\frac{1}{2}z^2}{(1-z)^3}, \tag{5.10}$$

The stability function $R^{(2)}$, as well as the internal stability function $R^{(1)}$ is L-acceptable. Thus, because of the fact that all the coefficient functions have a zero at infinity, the method is internally S-stable.

The given methods were applied to three stiff, non-linear problems which are well known in literature (see Enright et al. [2]). These problems all have real eigenvalues. We emphasize that the results continue to hold in case of non-real eigenvalues. The problems read:

I. $y_1' = y_3 - 100 y_1 y_2,$
$y_2' = y_3 + 2y_4 - 100 y_1 y_2 - 2.10^4 y_2^2,$
$y_3' = 100 y_1 y_2 - y_3,$
$y_4' = 10^4 y_2^2 - y_4,$
$y_1(0) = y_2(0) = 1,\ y_3(0) = y_4(0) = 0,\ 0 \leqq x \leqq 20,$
reference solution at $x = 20$:
$y_1 = 0.6397604447, \quad y_2 = 0.5630850708_{10} - 2,$
$y_3 = 0.3602395553, \quad y_4 = 0.3170647970.$

II. $y_1' = 0.01 - (1 + (y_1 + 1000)\,(y_1 + 1))\,(0.01 + y_1 + y_2),$
$y_2' = 0.01 - (1 + y_2^2)\,(0.01 + y_1 + y_2),$
$y_1(0) = y_2(0) = 0,\ 0 \leqq x \leqq 100,$
reference solution at $x = 100$:
$y_1 = -0.99164207, \quad y_2 = 0.98333636.$

III. $y_1' = -0.013 y_2 - 1000 y_1 y_2 - 2500 y_1 y_3,$
$y_2' = -0.013 y_2 - 100 y_1 y_2,$
$y_3' = -2500 y_1 y_3,$
$y_1(0) = 0,\ y_2(0) = y_3(0) = 1,\ 0 \leqq x \leqq 50,$
reference solution at $x = 50$:
$y_1 = -0.189338654_{10} - 5, \quad y_2 = 0.597654698,$
$y_3 = 1.402343409.$

The main purpose of this section is to compare the stability of the various methods for non-linear problems. Therefore, we confined ourselves to the following simple stepsize strategies A and B, given by

$$A(h^{(1)}, h^{(2)}):\ h_n = \begin{matrix} h^{(1)}, & x < 0.1, \\ h^{(2)}, & x \geqq 0.1, \end{matrix}$$

where $h^{(1)} \ll h^{(2)}$, and

$$B(h):\ h_n = h.$$

The numbers h, $h^{(1)}$ and $h^{(2)}$ are specified at the tables. In these tables we give for each j-th component the number of correct digits $sd_j = -\log_{10}$ (absolute error), at the endpoint of the given integration interval. In the tables, the letter u refers to an unstable result. The calculations have been performed on a Cyber 73–28 computer using 14 significant digits.

Table 5.1. Results for problem I

Strategy	A (0.01, 1)			A (0.001, 0.1)			B (1)			B (0.1)		
Method	I	II	III	I	II	III	I	II	III	I	II	III
sd_1	u	u	>9	u	u	>9	u	u	6.4	u	u	>9
sd_2	u	u	>9	u	u	>9	u	u	8.0	u	u	>9
sd_3	u	u	>9	u	u	>9	u	u	6.4	u	u	>9
sd_4	u	u	>9	u	u	>9	u	u	6.0	u	u	8.7

Table 5.2. Results for problem II

Strategy	A (0.01, 1)			A (0.001, 0.1)			B (1)			B (0.1)		
Method	I	II	III	I	II	III	I	II	III	I	II	III
sd_1	u	4.3	4.4	5.6	5.8	5.4	u	3.6	4.2	u	4.6	5.2
sd_2	u	4.6	4.0	5.9	5.9	5.5	u	3.5	3.9	u	4.5	5.2

Table 5.3. Results for problem III

Strategy	A (0.01, 1)			A (0.001, 0.1)			B (1)			B (0.1)		
Method	I	II	III	I	II	III	I	II	III	I	II	III
sd_1	u	10.3	10.3	9.9	12.3	12.3	u	10.0	10.3	9.9	12.0	12.3
sd_2	u	5.7	4.9	7.6	7.7	6.9	u	4.4	4.9	6.8	6.4	6.9
sd_3	u	5.7	4.9	7.6	7.7	6.9	u	4.4	4.9	6.8	6.4	6.9

The results of these computations indicate that the stability properties, derived for the *S*-stability test-model, carry over to a certain extent to non-linear systems. When applied to stiff, non-linear systems, *S*-stable methods from our class exhibit a much better stability behaviour than *A*-stable methods. Moreover, we may also conclude that for strongly non-linear problems, internally *S*-stable methods are superior to the *S*-stable ones. These conclusions are strenghtened by other experiments reported in Verwer [10]. Finally, it is also of interest to observe that in cases where all methods were stable roughly the same accuracy was obtained.

Acknowledgements. The author wishes to thank P. J. Van der Houwen and M. van Veldhuizen for their useful comments during the course of the investigation, and C. den Heyer for carefully reading the manuscript.

References

1. Dahlquist, G. G.: A special stability problem for linear multistep methods, BIT **3**, 27–43 (1963)
2. Enright, W. H., Hull, T. E., Lindberg, B.: Comparing numerical methods for stiff systems of ODEs, Techn. Report No. 69, Dept. of Comp. Sc., University of Toronto (1974)
3. Lambert, J. D.: Computational methods in ordinary differential equations. London: John Wiley and Sons 1973
4. Prothero, A., Robinson, A.: On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. Math. Comp. **28**, 145–162 (1974)
5. Rosenbrock, H. H.: Some general implicit processes for the numerical solution of differential equations. Comput. J. **5**, 329–330 (1963)
6. Van der Houwen, P. J.: Explicit and semi-implicit Runge-Kutta formulas for the integration of stiff equations. Report TW 132, Mathematisch Centrum, Amsterdam (1972)

7. Van der Houwen, P. J.: Construction of integration formulas for initial value problems. Amsterdam: North-Holland Publishing Company 1976
8. Van Veldhuizen, M.: Convergence of one-step discretizations for stiff differential equations (Thesis), Mathematical Institute, University of Utrecht, Netherlands (1973)
9. Verwer, J. G.: *S*-stability and stiff-accuracy for two classes of generalized integration methods. Report NW 15/75, Mathematisch Centrum, Amsterdam (1975)
10. Verwer, J. G.: Internal *S*-stability for generalized Runge-Kutta methods. Report NW 21/75, Mathematisch Centrum, Amsterdam (1975)
11. Verwer, J. G.: On generalized linear multistep methods with zeroparasitic roots and adaptive principal root. Numer. Math., **27**, 143–155 (1977)

Dr. J. G. Verwer
Mathematisch Centrum
2e Boerhaavestraat 49
Amsterdam (Oost), The Netherlands

A CLASS OF STABILIZED THREE-STEP RUNGE-KUTTA METHODS FOR THE NUMERICAL INTEGRATION OF PARABOLIC EQUATIONS *)

by

J.G. VERWER

ABSTRACT

A class of explicit three-step Runge-Kutta methods is discussed for the numerical solution of initial value problems for systems of ordinary differential equations. Attention is focussed on systems which originate from parabolic partial differential equations by applying the method of lines. New stabilized schemes of first and second order are presented. Some numerical examples are discussed.

*) To appear in the Journal of Computational and Applied Mathematics.
The present paper has been copied from the Mathematical Centre Report (prepublication) NW 39/77

1. INTRODUCTION

This paper deals with the construction of a class of stabilized explicit integration formulas for the numerical solution of systems of ordinary differential equations

$$y' = f(y) \tag{1.1}$$

Attention is focussed on systems which originate from semi-discretization of parabolic differential equations. Throughout this paper, it is assumed that the eigenvalues of the Jacobian matrix, say $J(y)$, of system (1.1) are spread out over a long narrow strip around the negative axis of the complex plane. For the majority of parabolic problems this assumption is satisfied (see Richtmeyer & Morton [4]).

A known advantage of explicit methods above implicit ones is their robustness, i.e. an explicit method can easily be applied to a very general class of problems. For example, system (1.1) may be linear or non-linear, it may originate from semi-discretization of a single or a system of parabolic equations in one or more dimensions. A known disadvantage of explicit methods, when compared with implicit ones, is that they necessarily possess a finite absolute stability region. This may impose a severe restriction to the stepsize. The main purpose of this paper is now to develop explicit methods possessing extended stability regions along the negative axis, so that the restrictions to the stepsize are reduced. A method possessing an extended stability region is called a stabilized method.

The formulas we discuss are three-step formulas belonging to the wide class of multistep Runge-Kutta methods, a class of integration methods which was originally discussed by Gear [1]. In the terminology of Gear such a method

is called a hybrid method. A thoroughly theoretical analysis of these methods has been given by Watt [11] . More recently, applications of multistep Runge-Kutta methods are discussed in Van der Houwen [7]. We shall also use the name multistep Runge-Kutta method of degree m, where m denotes the number of function evaluations. The discussion is confirmed to methods of order one and two. For a great deal of the problems of type (1.1), which occur in practice, methods of low order are valuable. The degree of the formulas constructed varies between two and twelve.

Stabilized one-step Runge-Kutta methods have already been discussed by Van der Houwen [6,7]. In the second reference he also pays attention to a special class of stabilized two-step methods. The change-over from one-step to two- or three-step schemes is of course meant to enlarge the real boundary of absolute stability, say β. To state the most important result of our investigations, we present new schemes for which holds:

$$\beta(m) \simeq 5.15\ m^2,\quad 2 \le m \le 12,\ \text{order} = 1,$$

$$\beta(m) \simeq 2.29\ m^2,\quad 2 \le m \le 12,\ \text{order} = 2.$$

For comparison, these boundaries are nearly three times larger than corresponding boundaries for stabilized one-step methods, provided the same damping properties are required.

2. ANALYSIS OF THE INVESTIGATED CLASS OF METHODS

The class of methods we consider may be represented by the following formula:

$$y_{n+1}^{(0)} = y_n,$$

$$y_{n+1}^{(1)} = (1-b_1)\ y_n + b_1 y_{n-1} + c_1 hf(y_{n-1}) + \lambda_{1,0}\ hf(y_n), \tag{2.1}$$

$$(2.1)\qquad y_{n+1}^{(j)} = (1-b_j)y_n + b_j y_{n-1} + c_j hf(y_{n-1}) + \lambda_{j,0}\, hf(y_n) +$$

$$+ \lambda_{j,j-1}\, hf(y_{n+1}^{(j-1)}),\ j = 2,\ldots,m;\ m \geq 2,$$

$$y_{n+1} = dy_{n+1}^{(m)} + (1-d)y_{n-2},\ n \geq 2.$$

The vector y_n always represents a numerical approximation to the analytical solution $y(x)$ at $x = x_n$. The points $x_j, j = n+1,\ldots, n-2$, denote the reference points of the three-step formula and h denotes the steplength, i.e. $h = x_{n+1} - x_n$, $n = 0,1,\ldots$. The steplength h is supposed to be constant. If $d = 1$ and $b_j = c_j = 0$, $j = 1,\ldots,m$, we have a one-step method which is discussed in Van der Houwen [6,7]. This one-step method may be used to provide the additional starting vectors y_1 and y_2. If $d = 1$, we have a two-step method which is discussed in Verwer [8].

There are two main reasons why we consider three-step formulas of the special class (2.1). Firstly, in order to reduce the storage requirements we only admit the derivatives $f(y_{n-1})$, $f(y_n)$ and $f(y_{n+1}^{(j-1)})$. By this choice our formulas need six arrays of storage. Secondly, in order to be able to apply the construction discussed in section 3.1, the vector y_{n-2} is not allowed to occur in the expressions for $y_{n+1}^{(j)}$ (cf. Verwer [9]).

As already noted in the introduction, this paper discusses the construction of stabilized formulas. As a consequence, we pay no attention to purely theoretical aspects. For a theoretical discussion of multistep Runge-Kutta methods the interested reader is referred to Watt [11], where also a convergence proof is given. For the usual definitions about convergence, consistency and stability we refer to Lambert [3].

2.1. CONVERGENCE AND CONSISTENCY CONDITIONS

The method is developed for the integration of partial differential equations. In a lot of applications of partial equations low order methods can be used succesfully. Therefore, we confine ourselves to methods of order $p = 1$ and $p = 2$. However, consistency conditions will be derived for $p \leq 3$. The third order conditions can then be used for a local error control in case of a second order method.

It is convenient to associate three-step method (2.1) with a non-linear difference operator

$$y_{n+1} - E[y_n, y_{n-1}, y_{n-2}]. \tag{2.2}$$

Now consider the initial value problem

$$y' = f(y),\ y(x_0) = y_0,\ x \geq x_0, \tag{2.3}$$

where f is a vector function of sufficient differentiability. Let y(x) be the solution of (2.3). Then the higher derivatives of y(x) can be expressed in terms of the function f and its derivatives. By using the tensor notation in Taylor's theorem for functions of several variables (see Henrici [2], p. 118), we obtain

$$y(x_{n+1}) - E[y(x_n), y(x_{n-1}), y(x_{n-2})] = \tag{2.4}$$

$$C_1 hf + C_2 h^2 f_j f^j + C_{31} h^3 f_j f^j_k f^k + C_{32} h^3 f_{jk} f^j f^k + O(h^4),$$

where

$$C_1 = 1 - \{d(-b_m+c_m+\lambda_{m,0}+\lambda_{m,m-1}) - 2(1-d)\},$$

$$C_2 = \frac{1}{2} - \{d(\frac{1}{2}b_m-c_m+\lambda_{m,m-1}(-b_{m-1}+c_{m-1}+\lambda_{m-1,0}+\lambda_{m-1,m-2})) + 2(1-d)\},$$

(2.5)
$$C_{31} = \frac{1}{6} - \{d(-\frac{1}{6}b_m+\frac{1}{2}c_m+\frac{1}{2}\lambda_{m,m-1}(b_{m-1}-2c_{m-1}+2\lambda_{m-1,m-2}(-b_{m-2}+$$

$$+c_{m-2}+\lambda_{m-2,0}+\lambda_{m-2,m-3}))) - \frac{8}{6}(1-d)\},$$

$$C_{32} = \frac{1}{6} - \{d(-\frac{1}{6}b_m+\frac{1}{2}c_m+\frac{1}{2}\lambda_{m,m-1}(-b_{m-1}+c_{m-1}+\lambda_{m-1,0}+\lambda_{m-1,m-2})^2) +$$

$$- \frac{8}{6}(1-d)\}.$$

Thus method (2.1) is consistent of order $p = 1$ if $C_1 = 0$, and consistent of order $p = 2$ if, in addition, $C_2 = 0$.

As is the case for linear multistep methods, a necessary condition for multistep Runge-Kutta methods to be convergent is the condition of zero-stability. In fact, the well-known convergence theorem for linear multistep methods, which states that the method is convergent if and only if it is zero-stable and consistent, applies for multistep Runge-Kutta methods. The condition, necessary for zero-stability of method (2.1) is (see Verwer [8]).

$$(2.6) \qquad d(c_m+\lambda_{m,0}+\lambda_{m,m-1}) \neq 0.$$

As is the case for linear multistep methods, it is recommended to use the left-hand side of (2.6) to normalize the error constants of the truncation error. We have chosen

$$(2.7) \qquad d(c_m+\lambda_{m,0}+\lambda_{m,m-1}) = 1,$$

which is assumed throughout this paper. If the left-hand side of (2.6) is chosen smaller than one, we have in fact a method of the Du Fort-Frankel type (see Richtmyer & Morton [4]).

2.2. ABSOLUTE STABILITY PROPERTIES

In order to investigate the absolute stability properties of method (2.1) it is applied to the linear test-model

$$(2.8) \qquad y' = \delta y, \ \delta \in \mathbb{C}.$$

This yields the recurrence relation

$$(2.9) \qquad y_{n+1} = d\,S(z)\,y_n + d\,P(z)\,y_{n-1} + (1-d)\,y_{n-2},$$

where $S(z)$ and $P(z)$ are polynomials of degree m in $z = h\delta$. For future reference, we shall call S and P the stability polynomials for method (2.1). Let us denote

(2.10) $$S(z) = \sum_{i=0}^{m} s_i z^i \text{ and } P(z) = \sum_{i=0}^{m} p_i z^i.$$

Then we have

$$s_0 = 1 - b_m,$$

$$s_1 = \lambda_{m,0} + \lambda_{m,m-1}(1-b_{m-1}),$$

(2.11)
$$s_i = \prod_{j=m-i+2}^{m} \lambda_{j,j-1}(\lambda_{m-i+1,0} + \lambda_{m-i+1,m-i}(1-b_{m-i})), \quad i = 2,\ldots,m-1,$$

$$s_m = \prod_{j=1}^{m} \lambda_{j,j-1},$$

and

$$p_0 = b_m,$$

$$p_1 = \lambda_{m,m-1} b_{m-1} + c_m,$$

(2.12)
$$p_i = \left(\prod_{j=m-i+1}^{m} \lambda_{j,j-1}\right) b_{m-i} + \left(\prod_{j=m-i+2}^{m} \lambda_{j,j-1}\right) c_{m-i+1}, \quad i = 2,\ldots,m-1,$$

$$p_m = \left(\prod_{j=2}^{m} \lambda_{j,j-1}\right) c_1.$$

The characteristic equation of the three-step recurrence (2.9) is

(2.13) $$\alpha^3 - d\, S(z)\, \alpha^2 - d\, P(z)\, \alpha - 1 + d = 0.$$

Before analyzing the stability of (2.9) by means of the characteristic roots of (2.13), it is convenient to express the consistency conditions for orders p = 1 and p = 2 into the first coefficients of S and P. This

can be done by using equations (2.5), (2.11) and (2.12), or alternatively, by substituting the second order Padé-approximation $1 + z + z^2/2$ to exp(z) into characteristic equation (2.13). The conditions are given below:

(2.14)

	$s_0 + p_0 = 1,$
$p = 1$	$s_1 - p_0 + p_1 = (3-2d)/d;$
$p = 2$	$s_2 + \frac{1}{2} p_0 - p_1 + p_2 = (-3/2+2d)/d;$

Thus these conditions are equivalent to the conditions $C_1 = 0$ for first order and $C_1 = C_2 = 0$ for second order, which are stated in the preceding section.

It is further convenient to express equation (2.7) in terms of d and p_0. By using the first of relations (2.5) and equality $p_0 = b_m$, we find

$$p_0 = \frac{2(d-1)}{d}. \tag{2.15}$$

It is our aim to develop stabilized formulas for parabolic equations. As the eigenvalues of the Jacobian matrix of such equations are generally real or almost real, it is of interest to develop formulas whose stability regions contain a considerable part of the negative axis. As a consequence, we state the following

STABILITY PROBLEM: Let z be negative and let $\alpha_i(z)$, $i = 1,2,3$, denote the roots of equation (2.13). Let ρ: $(-\infty,0) \to (0,1)$, $\rho(0) = 1$, be given and assume that relation (2.15) is satisfied. Then determine the coefficients s_i, p_i, $i = 0,\ldots,m$, in such a way that $\max_i \mid \alpha_i(z) \mid \leq \rho(z)$, $z \in [-\beta,0]$, β maximal, where it is assumed that $p = 1$ and $p = 2$, respectively.

The function ρ is introduced in order to obtain a strong damping for the higher harmonics. Moreover, ρ may be considered as an aid to construct a method of which the absolute stability region contains a narrow strip around the negative axis. If $\rho(z)$, $z < 0$, is not too close to 1, it is immediately clear that we can find such a region. The boundary β is the real boundary of absolute stability.

The construction of approximate solutions to the optimization problem is discussed in section 3.1. In section 3.1 we also give the results and show some absolute stability regions.

2.3. INTERNAL STABILITY PROPERTIES

An important concept for stabilized methods of the Runge-Kutta type is the concept of internal stability (see Van der Houwen [7], section 2.6.10). Internal stability deals with the propagation of round-off errors during a single integration step. For Runge-Kutta methods, possessing a large degree and a large stability boundary, this propagation may be

considerable and may easily influence the local accuracy. Van der Houwen analyzes the internal stability for the class of one-step methods which is contained in class (2.1). He defines a so-called internal stability function, i.e. a function which approximately controls the propagation of round-off errors during a single step. It turns out that method (2.1) possesses the same internal stability function.

Let $\rho_{n+1}^{(j)}$ denote the local error entering at stage j of the Runge-Kutta process. Let $\varepsilon_{n+1}^{(j)}$ denote the accumulated local error at stage j. Further, let $\bar{y}_{n+1}^{(j)}$ denote the perturbed $y_{n+1}^{(j)}$. Instead of (2.1) we then have the process

$$\bar{y}_{n+1}^{(0)} = y_n,$$

$$\bar{y}_{n+1}^{(1)} = (1-b_1)y_n + b_1 y_{n-1} + c_1 hf(y_{n-1}) + \lambda_{1,0} hf(y_n) + \rho_{n+1}^{(1)},$$

$$\bar{y}_{n+1}^{(j)} = (1-b_j)y_n + b_j y_{n-1} + c_j hf(y_{n-1}) + \lambda_{j,0} hf(y_n) +$$

$$+ \lambda_{j,j-1} hf(\bar{y}_{n+1}^{(j-1)}) + \rho_{n+1}^{(j)}, \quad j = 2,\ldots,m, \quad m \geq 2,$$

$$\bar{y}_{n+1} = d\ \bar{y}_{n+1}^{(m)} + (1-d)\ y_{n-2}, \quad n \geq 2.$$

The errors $\varepsilon_{n+1}^{(j)} = \bar{y}_{n+1}^{(j)} - y_{n+1}^{(j)}$ then satisfy:

$$\varepsilon_{n+1}^{(1)} = \rho_{n+1}^{(1)},$$

$$\varepsilon_{n+1}^{(j)} = \lambda_{j,j-1}\ h[f(y_{n+1}^{(j-1)} + \varepsilon_{n+1}^{(j-1)}) - f(y_{n+1}^{(j-1)})] + \rho_{n+1}^{(j)}, \quad j = 2,\ldots,m,$$

$$\varepsilon_{n+1} = d\ \varepsilon_{n+1}^{(m)},$$

where $\varepsilon_{n+1} = \bar{y}_{n+1} - y_{n+1}$. By assuming that the Jacobian J(y) is slowly varying during one step, there approximately holds

$$\varepsilon_{n+1}^{(j)} \simeq \lambda_{j,j-1}\, hJ(y_n)\, \varepsilon_{n+1}^{(j-1)} + \rho_{n+1}^{(j)},\quad j = 2,\ldots,m.$$

After some elementary calculations we then arrive at the estimate

$$\|\varepsilon_{n+1}\| \le \Big[d + \sum_{k=1}^{m-1} d \prod_{j=m+1-k}^{m} |\lambda_{j,j-1}|\, \|(hJ(y_n))^k\|\Big] \max_{1\le k\le m} \|\rho_{n+1}^{(k)}\|,$$

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

Following Van der Houwen we now define the internal stability function

$$Q(z) = d + \sum_{k=1}^{m-1} d \prod_{j=m+1-k}^{m} |\lambda_{j,j-1}|\, |z|^k. \tag{2.16}$$

In case of a normal matrix $J(y_n)$ there holds

$$\|\varepsilon_{n+1}\| \le Q(h\sigma(J(y_n))) \max_{1\le k\le m} \|\rho_{n+1}^{(k)}\|,$$

where σ denotes the spectral radius. Consequently, in actual computation the steplength h should at least satisfy the internal stability condition

$$Q(h\sigma(J(y_n))) \le \frac{\text{tolerance}}{\text{arithmetic precision}}, \tag{2.17}$$

where tolerance stands for the maximal local truncation error allowed. However, when cancellation of digits appears, this condition may even be too optimistic.

The significance of the internal stability condition becomes clear when one realizes that Q is a strongly increasing function in its argument, and that for stabilized methods large arguments occur. Values of $Q(\beta)$ will be given in section 3.2.

The significance of condition (2.17) is corroborated by practical experiments. It turns out that when (2.17) is satisfied the internal stability is generally under control. A numerical experiment illustrating the significance of the internal stability function is discussed in section 3.2.

3. CONSTRUCTION OF THE ALGORITHMS

In section 3.1 we discuss a heuristic solution technique which yields approximate solutions to the stability problem stated in section 2.2. Once the parameter d and the coefficients s_i and p_i are determined, it is easy to derive parameters for a three-step scheme. A class of schemes of first and second order is presented in section 3.2.

3.1. A SOLUTION TECHNIQUE FOR THE STABILITY PROBLEM

In order to save space we will confine ourselves to the main features of the technique. For details we refer to Verwer [9].

Suppose a damping function $\rho(z)$ is given. Then define $\alpha = \rho\xi$ and substitute into equation (2.13). This yields a cubic equation in ξ:

$$\rho^3\xi^3 - dS\rho^2\xi^2 - dP\rho\xi - (1-d) = 0. \tag{3.1}$$

Let

$$\xi = \frac{1 + \eta}{1 - \eta}, \tag{3.2}$$

which maps the interior of the unit circle $|\xi| = 1$ into the half-plane $\mathrm{Re}(\eta) < 0$. Substitution of (3.2) into (3.1) yields a cubic equation in η:

$$a_0\eta^3 + a_1\eta^2 + a_2\eta + a_3 = 0, \tag{3.3}$$

where

$$\begin{aligned} a_0 &= \rho^3 + dS\rho^2 - dP\rho + 1 - d, \\ a_1 &= 3\rho^3 + dS\rho^2 + dP\rho - 3(1-d), \\ a_2 &= 3\rho^3 - dS\rho^2 + dP\rho + 3(1-d), \\ a_3 &= \rho^3 - dS\rho^2 - dP\rho - (1-d). \end{aligned} \tag{3.4}$$

Sufficient conditions for the roots of (3.1) to lie inside or on the unit circle can be obtained by applying the Routh-Hurwitz criterion to (3.3) (see Lambert [3]). These conditions read:

$$a_i \geq 0, \quad i = 0,\ldots,3; \quad a_1a_2 - a_0a_3 \geq 0. \tag{3.5}$$

Observe that without the equality signs conditions (3.5) are necessary for the roots of (3.1) to lie inside the unit circle. In terms of S, P, d and ρ conditions (3.5) give:

$$\rho S - P \geq \frac{d - 1 - \rho^3}{d\rho},$$

$$\rho S + P \geq \frac{3(1-d) - 3\rho^3}{d\rho},$$

$$-\rho S + P \geq \frac{3(d-1) - 3\rho^3}{d\rho}, \tag{3.7}$$

$$-\rho S - P \geq \frac{1 - d - \rho^3}{d\rho},$$

$$\frac{1 - d}{\rho^2} S + P \geq \frac{(1-d)^2 - \rho^6}{d\rho^4}.$$

The problem stated in section 2.2 thus reads: Let the function ρ be given and let condition (2.15) be satisfied. Then determine the coefficients s_i and p_i, compatible to an imposed order of accuracy, in such a way that (3.7) is satisfied for $z \in [-\beta,0]$, β maximal.

The technique is based on the following heuristic idea: Suppose the parameter d is fixed beforehand. Then discretize the variable z on an interval $[-\overline{\beta},0]$, i.e. define points $z_j = j\Delta z$, $\Delta z = \overline{\beta}/N$, $j = 1,\ldots,N$, where N is prescribed. Next replace the five non-linear inequalities by a system of linear inequalities by substituting $z = z_j$, $j = 1,\ldots,N$. After adding $4 + 2p$ inequalities associated with consistency conditions (2.14) and relation (2.15), we arrive at the system

$$AX \geq C, \tag{3.8}$$

A being a $(5N+2p+4) * (2m+2)$ matrix, X being a $2m + 2$-vector of unknowns

$p_i, s_i, i = 0,\ldots,m$, and C being the $5N + 2p + 4$-vector of right-hand sides. If $\overline{\beta} \leq \beta$, β being the optimal real stability boundary, a feasible solution to (3.8) must exist. On the other hand, if $\overline{\beta} > \beta$ and N large enough, a feasible solution cannot exist. Such a feasible solution is easy to determine by using a linear programming method. Summarizing, once the optimal d is known, an approximate and almost optimal solution is easy to determine by solving a sequence of linear programming problems, e.g. by performing bisection on $\overline{\beta}$.

In actual calculation it is recommended to expand the polynomials S and P in orthogonal polynomials in order to prevent numerical difficulties for higher values of m. The vector X of unknowns then consists of the coefficients of the polynomial expansions.

Another remark of practical interest is the following. In order to satisfy (3.7) for arguments z between points z_j, it is necessary to choose N rather large. As a consequence, the number of constraints of (3.8) is much larger than the number of variables. With regard to computational efficiency, it is then more effective to solve a linear programming problem belonging to the transposed of (3.8). In order to realize this, consider problem

$$\text{(3.9)} \qquad \min B^T X, \; B = [1,\ldots,1]^T,$$

subject to

$$\text{(3.10)} \qquad AX \geq C, \; -\infty \leq X_i \leq \infty,$$

and its dual problem

(3.11) $\max C^T Y,$

subject to

(3.12) $A^T Y = B,\ Y_i \geq 0.$

From the foregoing it is clear that we are primarily interested in the existence or non-existence of a feasible solution to (3.8). Well, according to the duality theorem, the dual solution to (3.11) - (3.12) is the primal solution to (3.9) - (3.10), which is a feasible solution to (3.8), provided system (3.8) has a feasible solution. Thus in actual calculation it is recommended to use (3.11) - (3.12) for the determination of the solution to (3.8) which is optimal with respect to β.

There remains to describe the determination of the parameter d. It is trivial that d is restricted to $0 < d < 2$, and the assumption is that the optimal d is independent of m. This assumption has been confirmed by practical experiments. The idea is then to determine an approximation to the optimal value of d by a numerical search technique for low values of m. Another assumption, also confirmed by practical experiments, is that $\beta(m) \simeq K\, m^2$, K constant. This means that the bisection process on $\overline{\beta}$ has to be performed only for some low values of m. For further details we refer to Verwer [9].

Using the heuristic solution technique described in this section, approximate solutions to the stability problem were computed for $2 \leq m \leq 12$ for the function

$$(3.13)\qquad \rho(z) = \begin{cases} 1, & -1.5 < z \le 0, \\ 0.85, & z \le -1.5. \end{cases}$$

In the neighbourhood of the origin no damping is prescribed, the consistency and zero-stability of the method will take care off. The restriction to $m \le 12$ will be explained in the next section. As a result of our calculations we found

$$(3.14)\qquad \begin{aligned} \beta(m) &\simeq 5.15\ m^2, \quad p = 1, \\ \beta(m) &\simeq 2.29\ m^2, \quad p = 2. \end{aligned}$$

These boundaries are nearly three times larger than corresponding boundaries of one-step methods with the same damping. As a consequence of the discretization, the continuous damping for $-\beta \le z \le -1.5$ is not exactly 0.85, but approximately 0.9. The resulting values of the coefficients s_i and p_i for $i = 0,\ldots,m$ and $m = 2,\ldots,12$ are given in VERWER [9]. The corresponding value of the parameter d then follows from (2.15).

In order to illustrate that the absolute stability regions $\{z \mid z \in \mathbb{C}, |\alpha_i(z)| < 1,\ i = 1,2,3\}$, belonging to the constructed stability polynomials S and P, contain a long narrow strip around the negative axis, four of such regions are given in fig. 3.1.

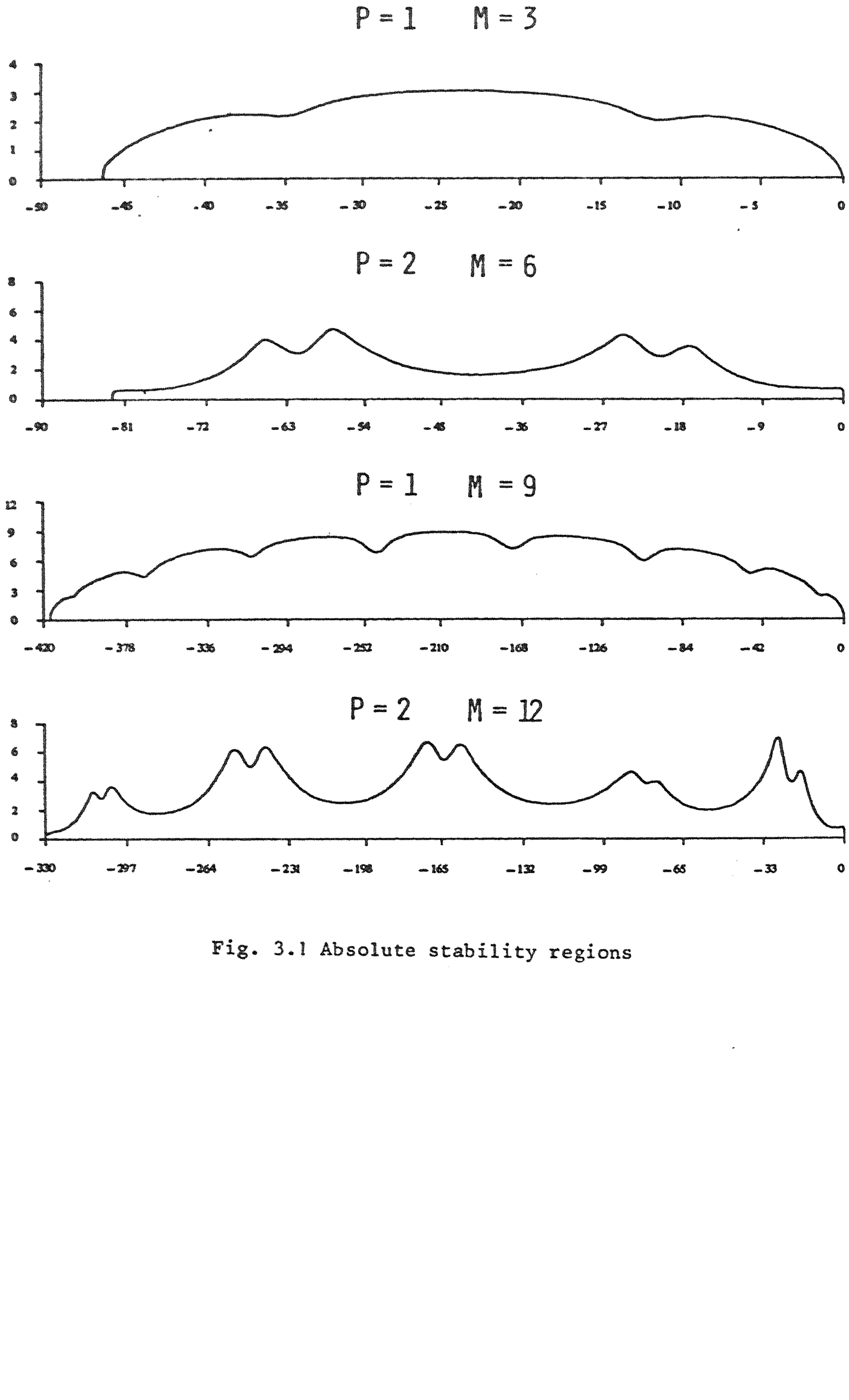


Fig. 3.1 Absolute stability regions

3.2. A CLASS OF FORMULAS OF FIRST AND SECOND ORDER

Once the coefficients s_i and p_i are given, integration parameters can be determined by using relations (2.11) - (2.12). From these relations it is easily seen that there exists more than one solution. We shall use this freedom by requiring that the local truncation error (2.4) satisfies a relation of the form

$$(3.15) \qquad y(x_{n+1}) - E[y(x_n), y(x_{n-1}), y(x_{n-2})] = C_{p+1}\, h^{p+1}\, y^{(p+1)}(x_n) + O(h^{p+2}),$$

where C_{p+1} is a constant. Representation (3.15) is convenient for a local error control based on interpolation with backward values. In the near future we intend to publish results of methods of type (2.1) with step-length and automatic error control based on such an interpolation (cf. Verwer [10]).

If $p = 1$, relation (3.15) is always satisfied (see expansion (2.4)). In order to obtain such a relation for $p = 2$, we have to require

$$(3.16) \qquad C_{31} = C_{32}.$$

Because of the equality $y''' = f_j f_k^j f^k + f_{jk} f^j f^k$, relation (3.15) is then satisfied with $C_3 = C_{31}$. Observe that for linear equations the error constants C_2 and C_3 are always determined by the coefficients

s_i and p_i. As a consequence of (3.16), the same holds for a non-linear equation too. It turns out that the error constants are almost independent of m, and thus also the accuracy of the schemes. The constants are rather large and approximately satisfy $C_2 \approx 1.27$, $C_3 \approx 0.44$.

Next we shall express the parameters of the schemes into the coefficients s_i and p_i. As there exists more than one solution, it is recommended to look for a positive, or almost positive solution in order to avoid a possible cancellation of digits. For p = 1 an almost positive solution is easily found, because of the fact that all s_i and p_i are positive. On the other hand, for p = 2 such a solution does not exist because of the fact that all p_i are negative, and all s_i are positive. Therefore, for p = 2, we select a solution which reduces the computational effort. To that end we set

$$b_i = 0,\ i = 1,\ldots,m-2;\quad \lambda_{i,0} = 0,\ i = 2,\ldots,m.$$

By using the consistency relations for p = 2 and relations (2.11) - (2.12), an elementary calculation then yields that condition (3.16) is satisfied, if and only if

$$(3.17)\qquad c_m = \frac{(1-\frac{1}{2}p_0)(p_1-2p_2+2p_3+2s_3)-(\frac{1}{2}+\frac{1}{4}p_0)^2}{2+p_1-2p_2+2p_3+2s_3}.$$

By performing some further elementary calculations, the remaining parameters can be solved from (2.11) - (2.12). Summarizing, we find the expressions:

$$b_i = 0, \quad i = 1,\ldots,m-2,$$

$$b_{m-1} = \frac{p_1 - c_m}{\lambda_{m,m-1}},$$

$$b_m = p_0;$$

$$c_i = \frac{p_{m+1-i}}{s_{m-i}}, \quad i = 1,\ldots,m-2,$$

$$c_{m-1} = \frac{p_2}{\lambda_{m,m-1}},$$

$$c_m = \frac{(1-\frac{1}{2}p_0)(p_1-2p_2+2p_3+2s_3)-(\frac{1}{2}+\frac{1}{4}p_0)^2}{2+p_1-2p_2+2p_3+2s_3}; \tag{3.18}$$

$$\lambda_{i,0} = 0, \quad i = 2,\ldots,m,$$

$$\lambda_{i,i-1} = \frac{s_{m+1-i}}{s_{m-i}}, \quad i = 1,\ldots,m-2,$$

$$\lambda_{m-1,m-2} = \frac{s_2}{\lambda_{m,m-1}},$$

$$\lambda_{m,m-1} = 1 - \tfrac{1}{2}p_0 - c_m.$$

In order to present the first and second order methods in a uniform way, we define the first order formulas also by expressions (3.18). The resulting parameter solution is almost positive.

Finally a remark about the range of the m-values. We restricted the m-values to $m \leq 12$, for we have to take into account the internal stability behaviour. To illustrate this, in table 3.1 we list the values $Q(5.15\ m^2)$, $m = 3,\ldots12$, for the first order formulas. The corresponding values $Q(2.29\ m^2)$ for the second order formulas are only slightly smaller. According to the in-

ternal stability condition (2.17), the smallest value of tolerance, allowed for a certain value of m, is then given by: tolerance = $Q(\beta)*$ arithmetic precision. Now suppose that the arithmetic precision is 14 digits, being a relevant machine precision nowadays. From table 3.1 it then follows that the local error may not be expected to be smaller than 10^{-5} if $m \geq 12$. Herewith it is assumed of course, that the maximally stable integration step is used. To our opinion a margin of 5 digits is acceptable, however it is recommended to choose the margin for the local accuracy not smaller.

As an illustration of the concept of internal stability we discuss an experiment for the simple linear system

$$
\begin{aligned}
y_1' &= (-2y_1+y_2+1)*10^4, \\
(3.19) \qquad y_j' &= (y_{j-1}-2y_j+y_{j+1})*10^4, \quad j = 2,\ldots,99, \\
y_{100}' &= (y_{99}-2y_{100}+1)*10^4.
\end{aligned}
$$

The Jacobian of (3.19) is well-known and a normal matrix. By prescribing the initial values $y_j(0) = 1$, $j = 1,\ldots,100$, we have the solution $y_j(x) \equiv 1$, $x \geq 0$. Thus in case of exact computations, i.e. no rounding errors occur, the integration schemes will yield the exact solutions, provided the parameters are exactly representable. In order to get an indication about the significance of the values $Q(\beta)$ we did perform one integration step with schemes of first and second order, for several values of m, using the steplength $h = \beta(m)/\sigma$. Here σ is equal to $4_{10}4$. The experiment has been carried out on a CDC 73/28 computer using an arithmetic precision of about 14 digits. Therefore the additional starting values are chosen as $1 + 10^{-14}*rn$, where rn denotes a random number between -1 and 1. In table 3.1 we have listed 10^{14} * error $_p(m)$, where

$$\text{error}_p(m) = \max_j (y_j - 1.0), \quad p = 1,2.$$

The results of table 3.1 clearly indicate that the behaviour of the propagation of rounding errors with increasing m is in accordance with the error analysis of section 2.3. For the first order formulas, the internal stability condition for equation 3.19) is somewhat too pessimistic. On the other hand, for the second order formulas the internal stability condition is too optimistic. Here cancellation of digits appears, which is due to the fact that the second order formulas possess positive and negative parameters. This means that second order formulas of a high degree must be used with some caution.

m	$Q(\beta)$	$10^{14} * \text{error}_1(m)$	$10^{14} * \text{error}_2(m)$
3	$.1_{10}3$	$.9_{10}1$	$.9_{10}2$
4	$.7_{10}3$	$.6_{10}2$	$.4_{10}3$
5	$.4_{10}4$	$.4_{10}3$	$.1_{10}5$
6	$.3_{10}5$	$.1_{10}4$	$.8_{10}5$
7	$.2_{10}6$	$.7_{10}4$	$.6_{10}6$
8	$.9_{10}6$	$.3_{10}5$	$.1_{10}7$
9	$.5_{10}7$	$.2_{10}6$	$.9_{10}7$
10	$.3_{10}8$	$.2_{10}7$	$.2_{10}9$
11	$.2_{10}9$	$.3_{10}7$	$.4_{10}10$
12	$.7_{10}10$	$.4_{10}8$	$.4_{10}11$

Tabel 3.1

4. NUMERICAL EXAMPLE

As already noted in section 3.2 in the near future we intend to publish results of schemes provided with automatic error, steplength and order control (cf. Verwer [10]). Intentionally we do not discuss these matters in this paper, because of the fact that constructing a formula and supplying an existing formula with control mechanisms leads to quite distinct problems. Therefore we discuss a numerical example where formulas are applied using a constant steplength.

We consider the non-linear initial-boundary value problem (from Sincovec & Madsen [5], but with a different initial value):

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u\frac{\partial u}{\partial x}\right) - u^2, \quad 0 \le x \le 1, \quad t \ge 0,$$

$$u(t,0) = 50, \quad u_x(t,1) = 1 - \sin(u), \quad t \ge 0, \tag{4.1}$$

$$u(0,x) = 50, \quad 0 \le x \le 1.$$

By using central differencing with respect to x the initial-boundary value problem (4.1) can be semi-discretized to the initial value problem

$$u_1' = [-(2+2(\Delta x)^2)\,u_1^2 + u_2^2 + 2500]/2(\Delta x)^2,$$

$$u_j' = [u_{j-1}^2 - (2+2(\Delta x)^2)u_j^2 + u_{j+1}^2]/2(\Delta x)^2, \quad j = 2,\ldots,N-1, \tag{4.2}$$

$$u_N' = [2u_{N-1}^2 - (2+2(\Delta x)^2)u_N^2 + 4\,\Delta x\, u_N(1-\sin(u_N))]/2(\Delta x)^2,$$

$$u_j(0) = 50, \quad j = 1,\ldots,N,$$

where $u_j(t)$ approximates $u(t,x_j)$, and where $x_j = j\Delta x$, $\Delta x = 1/N$, N prescribed.

When applied to semi-discretized problems such as (4.2), the step length for a stabilized explicit formula is usually limited by stabili Thus to apply such a formula efficiently a rough overestimate of $\sigma(J(y$ necessary. In fact, a program embodying an explicit method for semi-di cretized problems will have to calculate $\sigma(J(y))$ and possess a control mechanism for it.

For equation (4.2) an estimation of the spectral radius σ is easi found by using elementary matrix theory. If $u_j(t) > 0$ for $t \geq 0$, the elements of the lower and upper diagonal of the tridiagonal Jacobian a positive. Thus, if $u_j(t) > 0$, the Jacobian has real eigenvalues for $t \geq 0$. Further, by applying Gershgorin's circle theorem a small investigation of the Jacobian yields

$$\sigma \simeq 4(\Delta x)^{-2} \max_j u_j(t).$$

At $t = 0$ we then have

$$\sigma \simeq 200(\Delta x)^{-2}. \tag{4.3}$$

For the calculations we assume that $\max_j u_j(t) \simeq 50$, $t \geq 0$. With this assumption, and the assumption that $u_j(t) > 0$, approximation (4.3) can used for all t, and for all t we have real eigenvalues. After the inte tion of (4.2) both assumptions are easily verified.

We have carried out two experiments with problem (4.2) for $\Delta x = 1/30$. For problem (4.2) no analytical solution is available. Therefore, a computed reference solution at some selected times and points is given in table 4.1. In both experiments we applied a second order three-step scheme and, for comparison, a second order one-step scheme. The one-step scheme is defined by (cf. section 2)

$$y_{n+1}^{(0)} = y_n,$$

$$y_{n+1}^{(j)} = y_n + \lambda_{j,j-1}\, h\, f(y_{n+1}^{(j-1)}),\ j = 1,\ldots,m, \tag{4.4}$$

$$y_{n+1} = y_{n+1}^{(m)}.$$

The integration parameters $\lambda_{j,j-1}$ are given by

$$\lambda_{j,j-1} = \bar{s}_{m+1-j}/\bar{s}_{m-j},\ j = 1,\ldots,m-2;\ \lambda_{m-1,m-2} = \tfrac{1}{2},\ \lambda_{m,m-1} = 1,$$

where $\bar{s}_j, j = 3,\ldots,m$, are coefficients of the strongly stable stability polynomial

$$\tilde{R}_m^{(2)}(z) = 1 + z + \tfrac{1}{2} z^2 + \bar{s}_3 z^3 + \ldots + \bar{s}_m z^m$$

of (4.4), which are listed in Van der Houwen [7, table 2.6.7']. The extrema of $\tilde{R}_m^{(2)}$ are bounded by 0.95 in the real stability interval.

Experiment I: The integration has been performed with schemes of degree 12 using the maximally stable steplength. Thus in this experiment we ignore any accuracy condition. Let h_1 and h_3 denote the steplength, and let β_1 and β_3 denote the real stability boundary of the one-step and three-step scheme, respectively. Then (cf. Van der Houwen [7, table 2.6.7'])

$$\beta_1 = 0.8 * 144 = 115.20,\ h_1 = \frac{115.20}{200} * (\Delta x)^2 = 0.00064,$$

and

$$\beta_3 = 2.29 * 144 = 329.76,\ h_3 = \frac{329.76}{200} * (\Delta x)^2 = 0.001832.$$

The integration is stopped as soon as $t \geq 0.1$. At about $t = 0.1$ the steady state solution, i.e. the solution of the related two-point boundary value problem, is obtained. The additional starting vectors for the three-step scheme are obtained from the computed reference solution. The start of the three-step scheme is counted as two integration steps. Then the three-step scheme needs 55 steps, and the one-step scheme needs 157 steps to reach $t = 0.1$. In table 4.2 we give relative errors with respect to the computed reference solution at some selected times and points. In order to calculate relative errors at the selected times, quadratic interpolation has been used between the solutions computed by the schemes. An error equal to zero means that after rounding the interpolated value is equal to the computed reference solution within the specified number of digits.

t \ x	0.0	0.2	0.4	0.6	0.8	1.0
.010	50.000	45.091	41.471	39.041	37.708	37.429
.025	50.000	44.506	40.253	37.262	35.577	35.229
.050	50.000	44.403	40.024	36.891	35.058	34.576
.100	50.000	44.383	39.979	36.816	34.952	34.442

Table 4.1. Reference solution for problem (4.2), $\Delta x = 1/30$.

	one-step					three-step				
t \ x	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
.010	$2_{10}{}^{-4}$	$1_{10}{}^{-4}$	$4_{10}{}^{-4}$	$2_{10}{}^{-4}$	$5_{10}{}^{-4}$	$4_{10}{}^{-3}$	$4_{10}{}^{-3}$	$6_{10}{}^{-4}$	$3_{10}{}^{-3}$	$9_{10}{}^{-3}$
.025	$2_{10}{}^{-5}$	$3_{10}{}^{-5}$	$8_{10}{}^{-5}$	$3_{10}{}^{-5}$	$9_{10}{}^{-5}$	$7_{10}{}^{-4}$	$1_{10}{}^{-3}$	$7_{10}{}^{-4}$	$2_{10}{}^{-3}$	$3_{10}{}^{-3}$
.050	0	0	0	$3_{10}{}^{-5}$	$6_{10}{}^{-5}$	$5_{10}{}^{-5}$	$2_{10}{}^{-4}$	$2_{10}{}^{-4}$	$4_{10}{}^{-4}$	$5_{10}{}^{-4}$
.100	0	0	0	0	0	0	0	0	$3_{10}{}^{-5}$	$3_{10}{}^{-5}$

Table 4.2. Results of experiment I.

From table 4.2 we conclude that the schemes yield the steady state solution at t = 0.1 with almost the same accuracy. In the initial phase of the integration the one-step scheme yields more accurate results than the three-step scheme. This is what we expect, as $h_3 \simeq 3h_1$. In order to get some indication about the accuracy behaviour of our three-step methods, when compared with stabilized one-step methods, we did another integration with two schemes using the same steplength. Before discussing this integration in experiment II, we observe that we only consider results of the time integrations. For the reference solution, given in table 4.1, belongs to the system of ordinary differential equations (4.2).

Experiment II: Problem (4.2) has been integrated with a one-step scheme and a three-step scheme with steplength h = 0.0005 over the interval [0,0.1]. As observed in section 3.2, the accuracy of the three-step schemes is independent of the degree. The same holds for the one-step schemes. Therefore it is allowed to select the degree m of the schemes in such a way that

(4.5) $\qquad h\,\sigma \leq \beta(m), \quad m \text{ minimal.}$

Let m_1 and m_3 denote the degree of the one-step and three-step scheme that satisfy (4.5), respectively. According to Van der Houwen [7, table 2.6.7'], there holds $\beta_1(10) \simeq 79.70$, $\beta_1(11) \simeq 96.66$. With the specified h and σ we then find $m_1 = 11$ and $m_3 = 7$.

The additional starting vectors for the three-step scheme are obtained from the computed reference solution. The start of the three-step scheme is again counted as two integration steps. Then the three-step scheme needs 1400, and the one-step scheme 2200 function evaluations to reach t = 0.1. Relative errors are listed in table 4.3 and are computed in the same way as in experiment I.

	one-step					three-step				
t \ x	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
.010	$2_{10}{}^{-5}$	$5_{10}{}^{-5}$	$3_{10}{}^{-5}$	$2_{10}{}^{-4}$	$5_{10}{}^{-5}$	$9_{10}{}^{-5}$	$2_{10}{}^{-4}$	$3_{10}{}^{-4}$	$5_{10}{}^{-4}$	$5_{10}{}^{-4}$
.025	0	0	0	$3_{10}{}^{-5}$	0	$2_{10}{}^{-5}$	$5_{10}{}^{-4}$	$1_{10}{}^{-4}$	$1_{10}{}^{-4}$	$2_{10}{}^{-4}$
.050	0	0	0	$3_{10}{}^{-5}$	$6_{10}{}^{-5}$	0	0	0	$3_{10}{}^{-5}$	$9_{10}{}^{-5}$
.100	0	0	0	0	0	0	0	0	0	0

Table 4.3. Results of experiment II.

From table 4.3 we conclude that the results of the one-step integration are more accurate than the results of the three-step integration. This is corroborated by other practical experiments. However, due to condition (4.5), in many situations the degree of the three-step scheme can be chosen smaller. With other words, the three-step scheme may integrate with a smaller step-length than the one-step scheme with the same computational effort. This

may lead to comparable, or even higher accuracy. As an illustration, problem (4.2) has been integrated another time with the three-step scheme of degree four using the steplength h = 1/5500 over the interval [0,.1]. This integration also costs 2200 function evalutions. The relative errors are listed in table 4.4.

t \ x	0.2	0.4	0.6	0.8	1.0
.010	0	$2_{10}{}^{-5}$	$5_{10}{}^{-5}$	$8_{10}{}^{-5}$	$8_{10}{}^{-5}$
.025	0	0	$3_{10}{}^{-5}$	0	$3_{10}{}^{-5}$
.050	0	0	0	0	$6_{10}{}^{-5}$
.100	0	0	0	0	0

Table 4.4. Results of the three-step scheme with m = 4.

Summarizing, when the steplength is completely determined by stability and not by accuracy conditions, the three-step schemes can integrate with stepsizes which are nearly three times larger than the stepsizes allowed for the one-step schemes. If the steplength is completely determined by accuracy, the one-step schemes will generally yield more accurate results than the three-step schemes when using the same steplength. With respect to computational efficiency however, we expect that in general the three-step schemes can compete with the one-step schemes when an accurate time integration is necessary.

Perhaps it is needless to say that the integration formulas discussed in this paper, can equally be applied to parabolic problems in two, or

possibly three dimensions. The only mathematical restriction is that the eigenvalues of the Jacobian of the system of ordinary differential equations, which exists by semi-discretization, are almost real. For a lot of problems, this means no restriction. A practical restriction on the use of stabilized explicit methods may arise when $\sigma(J(y))$ is extremely large. In spite of the relatively large stability boundaries, very small timesteps are then required to maintain stability. In particular, this disadvantage applies when the integration has to be executed over a relatively large interval. In such a situation it may be necessary to apply an unconditionally stable method. For one-dimensional problems this is easy to realize (see e.g. Sincovec & Madsen [5]). However, for multi-dimensional problems implicit methods are in general difficult to implement, whereas explicit methods are very easy to implement.

ACKNOWLEDGEMENTS. The author wishes to thank Prof. P.J. Van der Houwen for his constructive comments during the preparation of this paper. The author is also grateful to Mr C. den Heyer for his careful reading of the manuscript.

REFERENCES

[1] C.W. Gear, Hybrid methods for initial value problems in ordinary differential equations, J. SIAM Numer. Anal., Ser. B, 2(1964), pp. 69-86.

[2] P. Henrici, Discrete variable methods in ordinary differential equations, John Wiley & Sons, New York, 1962.

[3] J.D. Lambert, Computational methods in ordinary differential equations, John Wiley & Sons, London, 1973.

[4] R.D. Richtmeyer & K.W. Morton, Difference methods for initial value problems, Interscience, New York, 1967.

[5] R.F. Sincovec & N.K. Madsen, Software for nonlinear partial differential equations, ACM Transactions on mathematical software, 1(1975), pp. 232-260.

[6] P.J. van der Houwen, Explicit Runge-Kutta formulas with increased stability boundaries, Numer. Math., 20(1972), pp. 149-164.

[7] P.J. van der Houwen, Construction of integration formulas for initial value problems, North-Holland Publishing Company, Amsterdam, 1976.

[8] J.G. Verwer, Multipoint multistep Runge-Kutta methods I: On a class of two-step methods for parabolic equations, Report NW 30/76, Mathematical Centre, Amsterdam, 1976.

[9] J.G. Verwer, Multipoint multistep Runge-Kutta methods II: The construction of a class of stabilized three-step methods for parabolic equations, NW 31/76, Mathematical Centre, Amsterdam, 1976.

[10] J.G. Verwer, An implementation of a class of stabilized, explicit methods for the time integration of parabolic equations, Mathematical Centre, Amsterdam, (to appear).

[11] J.M. Watt, The asymptotic discretization error of a class of methods for solving ordinary differential equations, Proc. Camb. Phil. Soc. 61 (1967), pp. 461-472.

AN IMPLEMENTATION OF A CLASS OF STABILIZED, EXPLICIT METHODS FOR THE TIME INTEGRATION OF PARABOLIC EQUATIONS *)

by

J.G. VERWER

ABSTRACT

The paper deals with an implementation of a class of explicit three-step Runge-Kutta methods for the numerical solution of initial value problems for systems of ordinary differential equations. The systems we have in mind do originate from parabolic partial differential equations by applying the method of semi-discretization. The underlying schemes are stabilized and of first and second order. The number of function evaluations per step varies between two and twelve. The implementation is provided with steplength, error and order control. A FORTRAN version of the implementation is available. Numerical results of this FORTRAN program, applied to two semi-discretized problems, are reported.

*) Copied from the Mathematical Centre Report (prepublication) NW 38/77.

1. INTRODUCTION

In this paper we discuss the implementation of a class of explicit methods to be used for the time integration of semi-discretized parabolic partial differential equations. The semi-discretized system of ordinary differential equations is supposed to be in the autonomous form

$$y' = f(y). \tag{1.1}$$

An important property, possessed by the major part of semi-discretized parabolic systems, is that the spectrum of the Jacobian matrix, say $J(y)$, is almost real, i.e. the eigenvalues are situated in a long narrow strip around the negative axis of the complex plane. This property is essential for the implemented class of methods. Therefore it must be assumed that the problems, to which our time integrator is applied, possess this property.

At the present time many numerical methods exist for solving time dependent partial differential equations (see RICHTMYER & MORTON [8]). When dealing with more than one dimension, the greater part of these methods are not so easy to apply and can only efficiently be implemented for narrow classes of problems. As a consequence, the development of mathematical software for wide classes of linear, and also non-linear, partial differential equations is still in a very early state (see SINCOVEC & MADSEN [11] for a list of references). This is in direct contrast to the development in the field of ordinary differential equations (see e.g. SHAMPINE & GORDON [10]). Very capable software exists for wide classes of non-linear ordinary differential equations. By way of the method of semi-discretization, we can make use of the developments in this field (e.g. steplength and error control) for the implementation of a time integrator.

In this connection stabilized, explicit integration formulas are suited, because of the fact that these formulas, when used in conjunction with semi-discretization, are easy to apply, and can be implemented for wide classes of linear and non-linear problems in one and more dimensions. The only mathematical restriction, to be posed for parabolic problems, is that the

spectrum of the Jacobian of the semi-discretized system is almost real. A practical restriction, with respect to the application of such methods, may arise when the spectral radius of J(y) is extremely large. In spite of the relatively large stability boundaries, the methods are then forced to integrate with very small steps, which may result in an excessive runtime. In such a situation it may be preferable to use an implicit method, which is unconditionally stable (see RICHTMYER & MORTON [8]).

The integrator discussed is based on three-step Runge-Kutta formulas of order one and two. These formulas are stabilized with respect to the real boundary of absolute stability. In fact, the stability regions are long narrow strips around the negative axis of the complex plane. The stabilization of the formulas is achieved by using extra evaluations of the function f per integration step. The number of function evaluations may vary between two and twelve. The analysis and construction of the formulas is discussed in VERWER [14]. In section 2 of this paper we shall give a short review of the theoretical aspects. Section 3 is devoted to the actual implementation. In this section simple mechanisms for the steplength, error and order control are discussed. In section 4 we discuss M3RK which is a FORTRAN program based on the implementation discussed in section 3. The last section of this paper is devoted to a discussion of some numerical results obtained with M3RK.

Finally it should be noted that an ALGOL 60 version of an implementation of stabilized, explicit one-step Runge-Kutta methods (cf. VAN DER HOUWEN [13]) has already been given by BEENTJES [6].

2. THE CLASS OF INTEGRATION FORMULAS

In this section we shortly discuss the underlying class of methods. The analysis and construction of the formulas is extensively discussed in VERWER [14,15,16], where one can also find further references.

Our class of three-step Runge-Kutta formulas may be represented as follows:

$$y_{n+1}^{(0)} = y_n,$$

$$(2.1)\qquad y_{n+1}^{(j)} = (1-b_j)y_n + b_j y_{n-1} + c_j h f(y_{n-1}) + \lambda_j h f(y_{n+1}^{(j-1)}), \quad j = 1,\ldots,m,$$

$$y_{n+1} = d y_{n+1}^{(m)} + (1-d)y_{n-2}, \quad m \geq 2, \quad n = 2,3,\ldots.$$

The vector y_n denotes the approximation to the analytical solution $y(x)$ at $x = x_n$. The points x_j, $j = n-2,\ldots,n+1$, denote the reference points of the three-step formula and h denotes the steplength, i.e. $h = x_{n+1} - x_n$. In the present section h is supposed to be constant. For the application of (2.1) the additional starting vectors y_1 and y_2 must be given. Formula (2.1) is called a three-step formula of degree m, m being the number of function evaluations per integration step.

When applied to the scalar test-model

$$(2.2)\qquad y' = \delta y,$$

scheme (2.1) yields the linear recurrence relation

$$(2.3)\qquad y_{n+1} = dS(z)y_n + dP(z)y_{n-1} + (1-d)y_{n-2},$$

where $S(z)$ and $P(z)$ are polynomials of degree m in $z = h\delta$. The stability of method (2.1) depends on the parameter d and the coefficients of the stability polynomials S and P. We have implemented schemes of order $p = 1$ and $p = 2$, with degree m satisfying $2 \leq m \leq 12$. The corresponding polynomials S and P are such that the absolute stability regions contain a long narrow strip around the negative axis. We have

$$(2.4)\qquad \beta_1(m) \approx 5.15\ m^2, \quad \beta_2(m) \approx 2.29\ m^2,$$

where $\beta_1(m)$ denotes the real boundary of absolute stability for the p-th order scheme of degree m. For real eigenvalues, the extrema of the amplification factors of (2.3) are bounded by about 0.9 in the stability interval. Because of this we have a strong damping for the higher harmonics.

The integration parameters of (2.1) are expressed in the parameter d and the coefficients of S and P.

They are determined in such a way that the principal local truncation error, LTE_p say, is given by (see VERWER [14])

$$
\begin{aligned}
LTE_1 &\approx C_2 h^2 y^{(2)}(x_n), \qquad C_2 \approx 1.27, \\
LTE_2 &\approx C_3 h^3 y^{(3)}(x_n), \qquad C_3 \approx 0.44.
\end{aligned} \tag{2.5}
$$

Observe that LTE_p does not depend on m. For $p = 1,2$ and $m = 2,\ldots,12$, the coefficients s_i of S and p_i of P are given in VERWER [14]. The parameters b_j, c_j and λ_j are given by

$$
\begin{aligned}
b_m &= p_0, \\
c_m &= \frac{(1 - \frac{1}{2}p_0)(p_1 - 2p_2 + 2p_3 + 2s_3) - (\frac{1}{2} + \frac{1}{4}p_0)^2}{2 + p_1 - 2p_2 + 2p_3 + 2s_3}, \\
\lambda_m &= 1 - \tfrac{1}{2}p_0 - c_m, \\
b_j &= 0, \qquad j = 1,\ldots,m-2, \\
b_{m-1} &= \frac{p_1 - c_m}{\lambda_m}, \\
c_j &= \frac{p_{m+1-j}}{s_{m-j}}, \qquad j = 1,\ldots,m-2, \\
c_{m-1} &= \frac{p_2}{\lambda_m}, \\
\lambda_j &= \frac{s_{m+1-j}}{s_{m-j}}, \qquad j = 1,\ldots,m-2, \\
\lambda_{m-1} &= \frac{s_2}{\lambda_m}.
\end{aligned} \tag{2.6}
$$

The parameter d is independent of m and is given by

$$
d = 1.375, \quad p = 1, \qquad d = 0.775, \quad p = 2. \tag{2.7}
$$

Finally we mention the concept of internal stability. Because of the relatively large degree and relatively large stability boundaries, we have to deal with an accumulation of rounding errors which appears per integration step. Especially for the higher degree formulas it can easily reduce the local accuracy. For a formula of degree m this accumulation is approximately governed by a so-called internal stability function, say $Q_{m-1}^{[p]}(z)$, which is a strongly increasing polynomial of degree m - 1 Let σ denote the spectral radius. Then the accumulation is generally under control if we adjust the steplength h and the degree m to the so-called internal stability condition

$$(2.8) \qquad Q_{m-1}^{[p]}(h\sigma(J(y_n))) \leq \frac{\text{maximal local truncation}}{\text{arithmetic precision}} .$$

In the program we shall use the values $Q_{m-1}^{[p]}(\beta_p(m))$. For future reference, the values $Q_{m-1}^{[1]}(\beta_1(m))$ are listed in table 2.1. The values for the second order schemes, which are used in the program, are defined by $Q_{m-1}^{[2]}(\beta_2(m)) = 10^2\, Q_{m-1}^{[1]}(\beta_1(m))$.

m	2	3	4	5	6	7	8	9	10	11	12
	$3_{10}{}^{1}$	$1_{10}{}^{2}$	$7_{10}{}^{2}$	$4_{10}{}^{3}$	$3_{10}{}^{4}$	$2_{10}{}^{5}$	$9_{10}{}^{5}$	$5_{10}{}^{6}$	$3_{10}{}^{7}$	$2_{10}{}^{8}$	$1_{10}{}^{9}$

Table 2.1 The values $Q_{m-1}^{[1]}(\beta_1(m))$.

Finally we note that scheme (2.1) needs six arrays of storage. For the actual implementation discussed in the next section we also use six arrays.

3. THE IMPLEMENTATION OF THREE-STEP RUNGE-KUTTA FORMULAS

When integrating time dependent partial differential equations by using the method of semi-discretization there arise two types of discretization errors, viz. the error due to the spatial discretization and the error due to the time integration. In general the first error can not be controlled. To our opinion it is nevertheless useful to supply a method for the time

integration with various control mechanisms, if possible. By doing this one relieves the task of the user of such an integrator. To support this opinion we make the following observation. Our methods are conditionally stable. When applied to a non-linear system, it may then happen that a sudden instability arises because of an increase of the spectral radius. When a method is supplied with error and steplength control, such a sudden instability is immediately detected and the steplength is decreased.

The most widely applied implementation technique for linear multistep methods is nowadays the Nordsieck technique (see GEAR [5]). This technique makes it is very easy to realize error, steplength and also order control. Compared with a Lagrange implementation, i.e. an implementation where the y- and y'-values are stored, a Nordsieck implementation is less efficient for large systems because of the higher overhead costs. Therefore we prefer the Lagrange implementation for our formulas. As we have to deal with a low order and with three-step formulas, this yields no particular problems.

The greater part of the ideas we apply are well known and extensively discussed in the literature (see e.g. GEAR [5] and SHAMPINE & GORDON [10]). We shall therefore omit details where possible, but still observe that (as usual) most of the ideas we apply are based partly on theoretical arguments, and partly on heuristics.

3.1. THE START OF THE PROCESS

The two additional starting vectors y_1 and y_2 are computed by means of a one-step Runge-Kutta scheme of order $p = 2$, which is also formulated as a three-step scheme by introducing zero-parameters. It is obtained from (2.1) by putting

$$d = 1,\ b_j = c_j = 0,\ \lambda_j = r_{m+1-j}/r_{m-j}, \qquad j = 1,\ldots,m;\quad 2 \le m \le 12, \tag{3.1}$$

where r_j, $j = 0,\ldots,m$, denote the coefficients of the corresponding m-th degree stability polynomial, say R_m. For $m = 2$ this polynomial is given by $R_2(z) = 1 + z + \frac{1}{2}z^2$. For $3 \le m \le 12$ this polynomial is chosen equal to the stabilized polynomial $\tilde{R}_m^{(2)}$ given by VAN DER HOUWEN [13, table 2.6.7']. The extrema of $\tilde{R}_m^{(2)}$ in its real interval of absolute stability, say $(-\tilde{\beta}_2(m),0)$, are bounded by 0.95. Observe that for $m = 2$ no specific damping properties are imposed. For $m = 2$ the absolute stability boundary is 2. For convenience we approximate the boundaries $\tilde{\beta}_2(m)$ with

(3.2) $$\tilde{\beta}_2(m) \approx 0.44\ m^2 + 0.03\ m^3.$$

If $m \neq 12$, these approximations are slightly smaller than the true boundaries of $\tilde{R}_m^{(2)}$.

The internal stability behaviour of the starting schemes is roughly the same as that of the three-step schemes. In particular, the values given in table (2.1) hold for the starting schemes.

In order to start the process we need an initial steplength, say h_{start}. This initial steplength should be related to the local tolerance, say TOL, which is specified by the user. We estimate h_{start} as follows. Let $\sigma_0 = \sigma(J(y_0))$, σ denoting the spectral radius, be given (if σ_0 is not available, it is estimated by the program as outlined in section 3.5). Let $\|\ .\ \|$ denote the divided Euclidean norm (i.e. Euclidean norm divided by the square root of the number of components). The idea is now to estimate $\frac{1}{2}\sigma_0^{-2} y^{(2)}(x_0)$ which represents the last Taylor term taken into account by the actual start formula, obtained with stepsize σ_0^{-1}. By relating this conservative estimation of the principal local truncation error of the start formula with TOL, we reasonably obtain a safe estimation of h_{start}. Following this idea h_{start} is then defined by

(3.3) $$h_{start} = \sigma_0^{-1} (\eta_t / \eta_e)^{1/2} / 10,$$

where

(3.4) $$\eta_t = TOL + TOL * \|y_0\|,$$
$$\eta_e = \sigma_0^{-1} \|f(y_o + \sigma_0^{-1} f(y_0)) - f(y_0)\|.$$

It is observed that in the root formula (3.3), which is used to extrapolate a new stepsize(cf. GEAR [5],p.156), the estimation of $\frac{1}{2}\sigma_0^{-2} y^{(2)}(x_0)$ is multiplied by 200 to obtain an extra safety margin.

In order to obtain absolute stability at the start of the process, the initial steplength must satisfy the stability condition

(3.5) $$h_{start}\sigma_0 \leq \tilde{\beta}_2(m_{max}),$$

where m_{max} denotes the maximal degree allowed with respect to internal stability. The estimation of σ_0 and m_{max} is discussed in section 3.5. If h_{start} does not satisfy (3.5), we put $h_{start} = \tilde{\beta}_2(m_{max})/\sigma_0$.

3.2. ESTIMATION AND CONTROL OF THE LOCAL ERROR

For the error control we use the local truncation error which is estimated by LTE_p (see(2.5)). For the estimation of LTE_p, p= 1,2, we apply the simple interpolation formulas (n > 2)

$$LTE_1 \approx \frac{c_2}{c_2 + \frac{1}{2}} [y_{n+1} - 2y_n + y_{n-1}], \quad c_2 = \frac{1}{2} - C_2,$$

(3.6)

$$LTE_2 \approx \frac{c_3}{c_3 + \frac{5}{6}} [y_{n+1} - 3(y_n - y_{n-1}) + y_{n-2}], \quad c_3 = \frac{1}{6} - C_3,$$

where, according to the definition of LTE_p, y_{n-1} and y_{n-2} are assumed to be approximations of a sufficiently high order to a local analytical solution at the points x_{n-1} and x_{n-2}, respectively.

The error criterion which is to be performed after each n-th integration step, n > 2, is the mixed criterion

$$\|LTE_p\| \leq TOL + TOL * \|y_{n+1}\|, \tag{3.7}$$

where TOL stands for a user specified tolerance parameter and $\|\cdot\|$ denotes the divided Euclidean norm. If (3.7) is satisfied the integration step is accepted, otherwise rejected.

During the start of the process, i.e. if n = 1,2, no error control is performed. This is justified by the conservative estimation of the initial steplength. If the third step fails however, all results are rejected and the process is restarted with h = h/10.

3.3. CHANGING THE STEPLENGTH

Before discussing the estimation and control of the steplength we first mention how we realize the change. Our integration formulas (2.1) are

developed for a fixed steplength. This means changing stepsize must be handled apart. In a Nordsieck implementation, changing stepsize is an interpolation-extrapolation process (see GEAR [5]). We also use interpolation-extrapolation to determine the new y-values.

Let h and αh denote the old and new steplength, respectively. The new values of y_{n-1} and y_{n-2} are then interpolated or extrapolated by means of the quadratic formula

$$(3.8)\quad y(x-\bar{\alpha}h) = \tfrac{1}{2}\bar{\alpha}(\bar{\alpha}-1)y(x-2h) + \bar{\alpha}(2-\bar{\alpha})y(x-h) + \tfrac{1}{2}(2-\bar{\alpha})(1-\bar{\alpha})y(x),$$

where $\bar{\alpha} = \alpha$ and $\bar{\alpha} = 2\alpha$, respectively. Formula (3.8) is applied for $p = 1$ and $p = 2$. The error introduced by (3.8) is of order three and is ignored at the estimation of LTE_2.

Applying (3.8) too frequently may lead to severe instabilities. Therefore we also use the rule of thumb: after a change of h at least 4 steps are performed with h fixed, provided a step is not rejected. We return to this point in the next section. The new values of y'_{n-1} are not computed by means of interpolation or extrapolation, but by using the derivative function f(y). To our experience this leads to a more stable process of steplength changing.

3.4. ESTIMATION AND CONTROL OF THE STEPLENGTH AND ORDER

The new steplength αh is estimated using the well known root formula. Let $\bar{\alpha}$ be defined by

$$(3.9)\qquad \bar{\alpha} = \left(\frac{TOL + TOL * \|y_{n+1}\|}{\|LTE_p\|} \right)^{\frac{1}{p+1}}.$$

Then we put

$$(3.10)\qquad \alpha = \begin{cases} \bar{\alpha}/2.0, & p = 1, \\ \bar{\alpha}/1.6, & p = 2. \end{cases}$$

The factors 2.5 and 1.3 are to provide a conservative estimate. In order to prevent marginal changes the change is not performed when $0.9 < \alpha < 1.1$. Moreover, in order to prevent an excessive decrease or increase of the steplength, α is bounded by 0.1 and 3.0, respectively. Because of the factors in (3.10), a decrease of the steplength is not necessarily due to a step failure.

The estimate of α is made when a step fails or at least 4 steps have been performed after the last change. Because of the fact that repeated rejections may be caused by severe errors, the process is interrupted if this happens three times in succession. After this interruption we make a restart as described in section 3.1.

Our formulas are explicit and thus conditionally stable. Therefore the steplength h is always bounded by

$$(3.11) \qquad h_{max}(p) = \beta(m_{max}) / \sigma,$$

$h_{max}(p)$ being the maximal steplength with respect to absolute stability. In (3.11), $\beta(m_{max})$ stands for $\beta_1(m_{max})$, $\beta_2(m_{max})$ or $\tilde{\beta}_2(m_{max})$.

Next we mention the implemented order control which is very simple and based on the fact that, in general, the steplength is bounded by stability requirements. As already observed the process is always started with a second order one-step scheme and a second order three-step scheme. During the process h normally increases until $h = h_{max}(2)$. If h reaches this value, 4 steps are performed with the second order scheme and $h = h_{max}(2)$, provided no step failure occurs. Then, α is estimated for $p = 1$. If this particular $\alpha < 1.1$, the process is continued with $h = h_{max}(2)$ and the current second order scheme. Otherwise the process is continued with a first order scheme, but, as a matter of caution, with $h = h_{max}(2)$. Then, the next time α is estimated, h is allowed to increase while $p = 1$. This specific check for an order decrease is made every four steps, provided $h = h_{max}(2)$.

If during a first order integration h becomes smaller than $h_{max}(2)$, p is reset to 2. It is observed that an order increase is not necessarily due to a step failure.

3.5. ESTIMATION AND CONTROL OF THE DEGREE AND SPECTRAL RADIUS

In order to control the propagation of local errors per integration step we want to satisfy condition (2.8). This is achieved by putting $m \leq m_{max}$, m_{max} being the maximal degree of the schemes, which satisfies (see table 2.1)

$$(3.12) \qquad Q_{m-1}^{[p]}(\beta_p(m)) \leq \frac{\text{TOL}}{\text{arithmetic precision}} \ .$$

For the three-step schemes m_{max} thus depends on p; there holds $m_{max}(2) \leq m_{max}(1)$. The maximal degree for the starting scheme is chosen equal to $m_{max}(2)$ of the three-step scheme. If the exceptional situation arises that $m_{max} < 2$ (the quotient of TOL and arithmetic precision is too small), the process is discontinued.

A property of stabilized methods is that the local truncation error of the formulas is approximately independent of m. Thus it is useful to mini - mize m with respect to the stability condition

$$(3.13) \qquad h\sigma \leq \beta(m)$$

for given h and σ, while $\beta(m)$ represents $\beta_1(m)$, $\beta_2(m)$ and $\tilde{\beta}_2(m)$, respectively. The degree m is computed in this way at the start of the process, at the change-over from a one-step to a three-step scheme, and further every time h or p is changed.

From the foregoing it is clear that we need an estimation of σ. Because of the fact that σ is used to determine $h_{max}(p)$ and to select m minimal with respect to (3.13), it must always be an upper estimation. Once σ is estimated and the system to be integrated is non-linear, it may be necessary to control the variation of σ. Thus we also need a control mechanism for the spectral radius. With respect to the estimation and control of σ we distinguish between 3 options, which option is chosen has to be specified by the user.

OPTION I. The user provides an estimation of σ. Especially for linear problems this is often easy to do.

OPTION II. The user does not provide an estimation. In this case σ is estimated by means of a power method which is adapted for general non-linear vector functions f(cf. LINDBERG [6]). This method may be described as follows.

Suppose $\sigma_0 = \sigma(J(y_0))$ is to be estimated. Let r_i be a random number from $[-\varepsilon,\varepsilon]$, $\varepsilon > 0$, and let v_0 be defined componentwise by

$$v_{0,i} = \begin{cases} y_{0,i}(1+r_i), & y_{0,i} \neq 0, \\ r_i, & y_{0,i} = 0. \end{cases} \tag{3.14}$$

Let $\varepsilon_{max} = \max(\varepsilon, \varepsilon \| v_0 \|_2)$. The adapted power method is then defined by the iteration

$$v_{j+1} = v_0 + \varepsilon_{max} \frac{f(v_j) - f(v_0)}{\| f(v_j) - f(v_0) \|_2}, \tag{3.15}$$

$$\rho_{j+1} = \frac{\| f(v_{j+1}) - f(v_0) \|_2}{\varepsilon_{max}},$$

where $v_1 = y_0$ and $j = 1,2,\ldots$. If f is linear, i.e. J(y) constant, then $\rho_j \to \sigma_0$. If f is non-linear, then choosing ε sufficiently small, J(y) is approximately constant in $S(v_0,\varepsilon)$. Thus for ε sufficiently small, ρ_j will converge to an accurate estimation of σ_0.

In our program we have set $\varepsilon = 10^4$APR, APR denoting the arithmetic precision (e.g. for CDC Cyber $\varepsilon = 10^{-10}$). The iteration (3.15) is stopped as soon as $|\rho_{j+1} - \rho_j| \leq 10^{-3}\rho_{j+1}$, provided $j \geq 4$. If this inequality is not satisfied within 50 iterations, the whole process is discontinued. In general, the process converges slowly because of the absence of a dominant eigenvalue. As a matter of safety we therefore put $\sigma_0 = 1.1\rho$, ρ being the last iterate.

In case of the second option the variation of the spectral radius is also controlled. We distinguish between two situations. Firstly, σ increases during the course of the integration and the process becomes unstable. The instability is immediately detected by the error control and results in a step failure. After a step failure we therefore simply reestimate σ, provided the failure was not in succession. Secondly, σ decreases during the course of the integration. To detect a decrease of σ we use an inaccurate estimation of σ, say σ^*, which is given by $\sigma^* = \rho_3$. The estimation σ^* is computed every 25 steps since the last estimation of σ or σ^*, and we decide to reestimate σ if σ^* has been decreased with more than ten percent.

We note that ρ_3 is in most cases a very rough estimation to σ. To our purpose this rough estimation suffices. At this place it is emphasized that random values are used in formula (3.14). Though these values are small, they may slightly influence σ and, in particular, ρ_3. As a consequence, one should use a random generator using a fixed generative value to be able to recover earlier obtained results (see also section 4).

OPTION III. The user does not provide an estimation, but decides that an initial estimation by means of the power method at the start suffices. Thus in this case no control on the variation of σ is performed. For linear problems it is clear that one chooses between the first and third option.

3.6. ALGORITHMIC CONNECTION BETWEEN THE CONTROL MECHANISMS

This short section is added in order to clarify the algorithmic connection between the mechanisms for controlling the order, the degree, the stepsize and the spectral radius. To this end an informal flowchart showing this connection is given in fig. 3.1. In this flowchart it is assumed that we choose option 2 for estimating and controlling the spectral radius. Because of the fact that during the start of the process no control is performed (see section 3.2), it is also assumed that we are already integrating with a first or second order three-step scheme (in particular it is assumed that $n > 3$). Note that at the start of the process the spectral radius is estimated, the maximal degree m_{max} is determined, the initial steplength h_{start} is estimated, the maximal steplength $h_{max}(p)$, $p = 1,2$, given by (3.11), is determined (see at the end of section 3.1), and the degree is minimized according to (3.13). A more detailed flowchart showing the complete implementation is given in the next section.

4. THE PROGRAM

In this section we describe a FORTRAN version of the implementation discussed in the previous section. The program consists of a main program and 11 small subprograms which are written to structure the program in order to make it more easily readable and easy to modify. We emphasize that the subprograms are meant to be called by the main program, which is the sub-

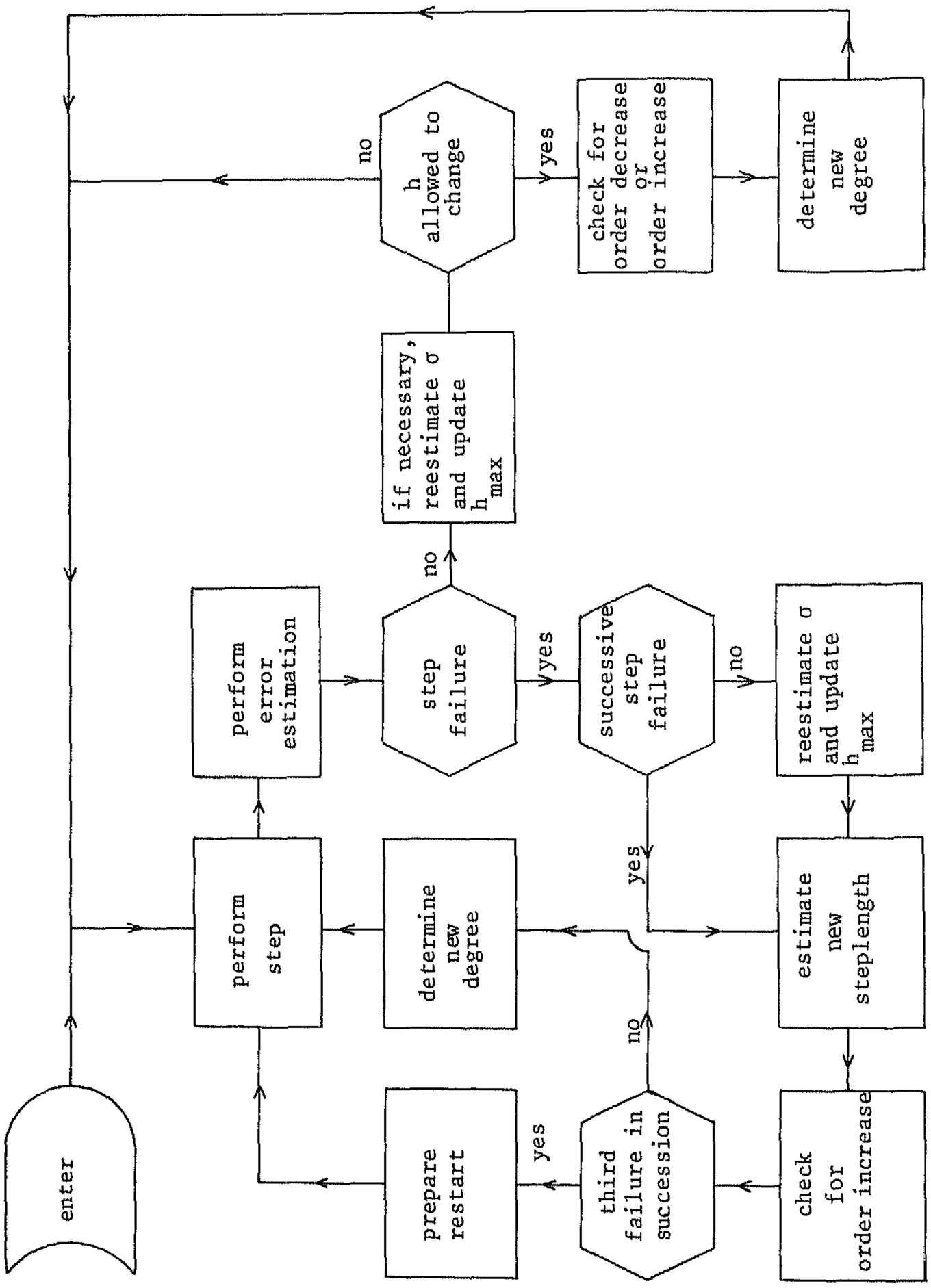


Fig. 3.1 Informal flowchart showing the algorithmic connection between control mechanisms.

routine M3RK, and thus can not stand on their own.

The program integrates a given problem from an initial value of x to a value of x which may be slightly beyond a user specified output point, say x_e. The desired solution at the output point x_e is always interpolated by means of the quadratic formula

$$(4.1) \qquad y(x_e) = \tfrac{1}{2}\mu(\mu-1)y(x-2h) + \mu(2-\mu)y(x-h) + \tfrac{1}{2}(\mu-1)(\mu-2)y(x),$$

where $\mu = (x-x_e) / h$. After a normal return of M3RK the parameters in the call list are ready to continue the integration. This means that when the user decides to continue the integration, he only needs to define a new output point x_e and call again. In fact, M3RK is written in such a way that the choice of output points does not influence the integration process itself.

All information to and from M3RK is passed through its parameters in the call list

```
M3RK (X, XE, N, H, HMIN, SIGMA, TOL, F, Y, Y1, Y2, YXE, DY,
      DY1, IFLAG, INFO)
```

X and XE represent the independent variable x and the output point x_e, respectively. N represents the number of differential equations of the system to be integrated. H an HMIN denote the steplength h and a smallest steplength, respectively.
SIGMA represents the spectral radius σ and TOL is the local tolerance parameter. F is the name of a subroutine defining the differential equations.
Y, Y1, Y2, YXE, DY and DY1 are arrays of length N which represent the solution vector at x, x − h, x − 2h, x_e and the derivative vector at x and x − h, respectively. We prefer the use of 6 arrays of length N instead of one of length 6N for clarity, and in order to avoid the overhead costs of pointers. When using one array this overhead is not negligeable for our class of formulas because of the high degree. A small disadvantage is of course a longer parameter list. IFLAG is an error flag and INFO an integer array of length 15, which is used to initialize the code, to pass information between M3RK and the subprograms, to pass information to the user about the status of

the integration, and finally to retain information for subsequent calls.

For specific information about input requirements and output we refer to the prologue of comments of M3RK, which further explains how to use the program (see appendix). Here we confine ourselves to the observation that for a first call the only input parameters are X, XE, N, TOL, F, Y, INFO(i), i = 1,2,3, while SIGMA is optional. For a subsequent call the only input parameter to be changed is normally XE. The user must always give a maximum for the number of evaluations of f(y) to be spent by M3RK. If this maximum number is reached while $x < x_e$, the process is interrupted and IFLAG is set equal to 1. In this situation the user has the possibility to continue the process in a simple way. The message IFLAG = 0 means that x_e is reached.

To give insight in the structure of the program we give a short list of the names and meanings of the subroutines which are called by M3RK:

HSTART computes the initial steplength according to section 3.1.

PARAM delivers the integration parameters according to expressions (2.6) and (3.1). PARAM contains a data-statement to store the coefficients of the stability polynomials S, P and $\tilde{R}_m^{(2)}$ into an internally declared array of length 440.

POWERM estimates the spectral radius (see section 3.5). If the computation fails IFLAG is set equal to 3.

MAXDEG evaluates the maximal degree m_{max} (cf. (3.12)). If $m_{max} < 2$, IFLAG is set equal to 2. MAXDEG contains a data-statement to store the 11 values given in table 2.1.

MINDEG evaluates the minimal degree satisfying the stability condition (3.3).

STEP contains the actual integrator and performs precisely one integration step with (2.1).

ESTIMA computes the local error bound and estimates the local error (see 3.7)).

NEWH delivers the new steplength and the factor α according to (3.10).

INTER1 performs the interpolation (3.8) and evaluates f(y(x-αh)).

INTER2 performs the interpolation (4.1) at the output point x_e.

SHIFT shifts the x- and y-variables and calculates a new derivative to prepare the next integration step.

In order to enlarge the portability of the program and to keep the structure as simple as possible we have avoided the use of common-statements.

As a consequence the subroutines are completely local, i.e. all information they need is passed through the parameter list. On purpose we do not discuss the various parameter lists. The meaning of the parameters should be immediately clear when reading M3RK which is extensively commented. Moreover, the subroutines called by M3RK are short and, as indicated above, directly associated to small parts of section 3. Thus these subroutines are easily verified by inspection. In figure 4.1 a flow chart is given which shows the structure of a complete program comprising a calling program, a user-supplied subroutine F, our main program M3RK and the subroutines called by M3RK. A downward sloping line from one box to another indicates that the lower program is called by the upper one. Observe however that F is always called via a parameter list.

A macroscopic flow chart of M3RK is given in fig. 4.2. In this flow chart k = 1 and k = 3 for a one-step and three-step formula, respectively: r denotes the number of successive rejected steps, n denotes the number of steps performed after start or restart, and s denotes the number of steps performed after an estimation, or check for an estimation of the spectral radius.

There remains to make some comments about the portability of the program. The whole package has been tested on a CDC 73/28 using 14 digits. It has been accepted by the PFORT Verifier (see RYDER [9]). The PFORT Verifier is a program which checks a FORTRAN program for adherence to PFORT, a portable subset of American National Standard FORTRAN. M3RK uses one machine dependent constant, namely the arithmetic precision represented by the internal variable EPR. POWERM contains the CDC system subprograms RANSET and RANF, constituting a random generator. RANF is the actual generator, while RANSET initializes the generative value of RANF. It is emphasized that replacing the CDC random generator slightly influences the results given in the next section.

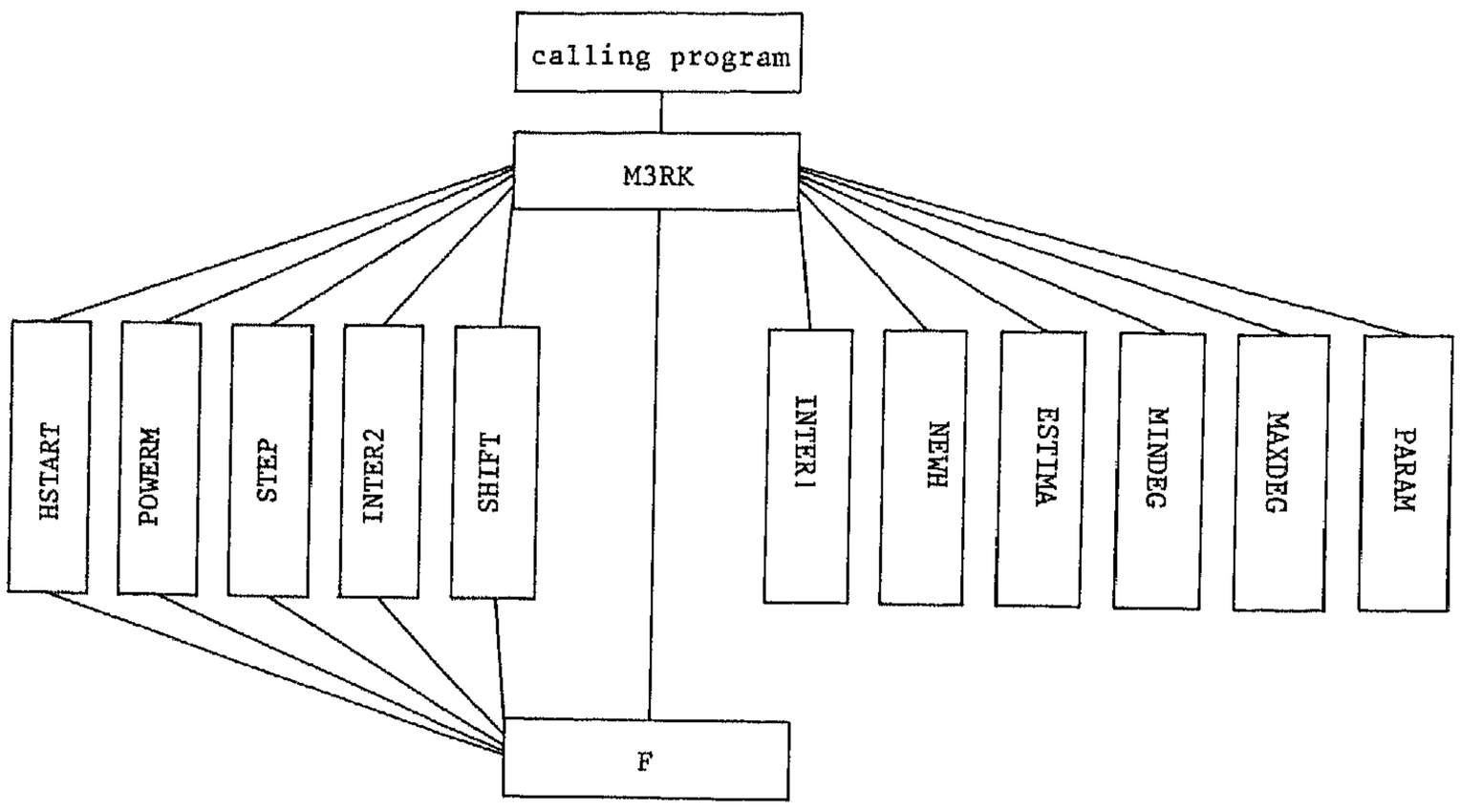


Fig. 4.1 Structure of a complete program

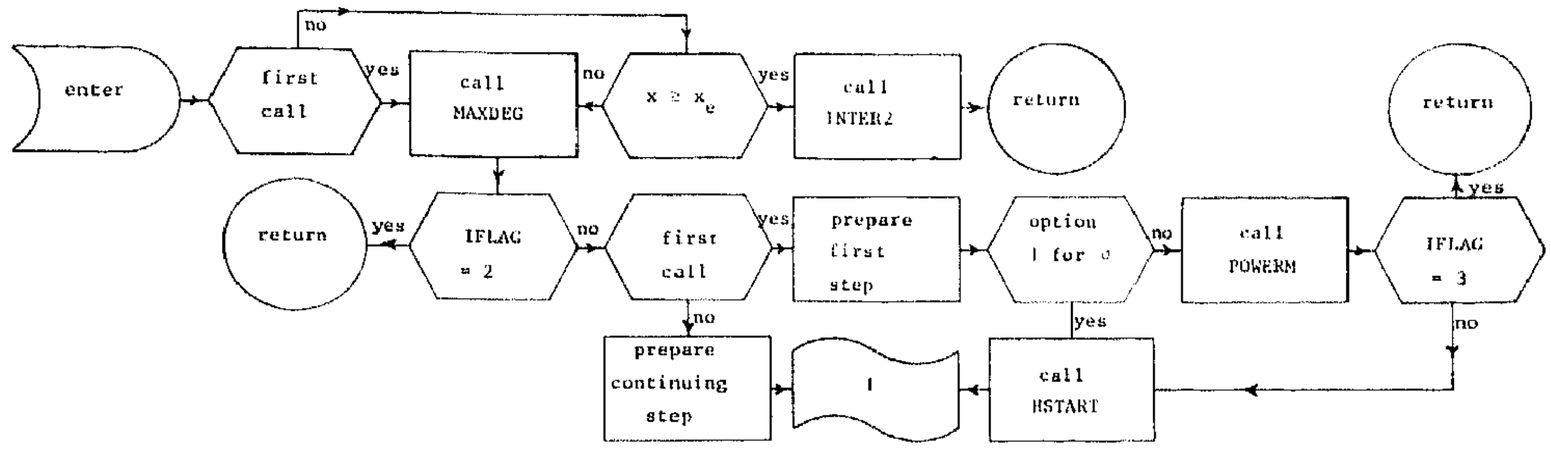


Fig. 4.2 Macroscopic flow chart of M3RK

5. NUMERICAL EXAMPLES

In order to test subroutine M3RK, it was applied to several problems. Two of these problems are discussed in this section. Both problems are non-linear and arise in practice. For both problems the semi-discretization has been performed by means of a continuous time Galerkin method (see e.g. DOUGLAS & DUPONT [4]).

5.1. A ONE-DIMENSIONAL SYSTEM OF TWO NON-LINEAR EQUATIONS

The first problem we consider is a one-dimensional system of two non-linear equations from electricity theory (cf. TE RIELE [12], section 3.2.6):

$$\begin{aligned} \frac{\partial u}{\partial t} &= \varepsilon\rho \frac{\partial^2 u}{\partial x^2} - g(u-v), \\ \frac{\partial v}{\partial t} &= \rho \frac{\partial^2 v}{\partial x^2} + g(u-v), \end{aligned} \tag{5.1}$$

where $0 \le x \le 1$, $g(z) = \exp(\frac{1}{3}\mu z) - \exp(-\frac{2}{3}\mu z)$, $\mu = 17.19$, $\varepsilon = 0.143$, $\rho = 0.1743$. The corresponding initial and boundary conditions read:

$$\begin{aligned} & u = 1, \quad v = 0, \quad 0 \le x \le 1, \quad t = 0, \\ & \frac{\partial u}{\partial x} = v = 0, \quad x = 0, \quad t > 0, \\ & u = 1, \ \frac{\partial v}{\partial x} = 0, \ x = 1, \ t > 0. \end{aligned} \tag{5.2}$$

Equation (5.1) was semi-discretized by means of a Galerkin method based on piecewise quadratic polynomials, and implemented to yield a purely explicit system of ordinary differential equations (cf. BAKKER [1]). It is beyond the scope of this paper to discuss this discretization technique. We confine ourselves therefore to the definition of the semi-discretized system obtained for the equidistant grid $\{x_i | x_i = (i-1)/(M-1),\ i = 1,\ldots,M\ ;M \text{ odd}\}$. Let $u_i(t) \approx u(x_i,t)$ and $v_i(t) \approx v(x_i,t)$. The equations for the u_i-components then are:

$$\dot{u}_1 = -\frac{1}{2}\varepsilon\rho(M-1)^2[7u_1 - 8u_2 + u_3] - g(u_1-v_1),$$

$$\dot{u}_{2i} = -\varepsilon\rho(M-1)^2[2u_{2i} - u_{2i-1} - u_{2i+1}] - g(u_{2i}-v_{2i})$$

$$i = 1,\ldots,\frac{M-1}{2}, \tag{5.3}$$

$$\dot{u}_{2i+1} = -\frac{1}{4}\varepsilon\rho(M-1)^2[14u_{2i+1} - 8(u_{2i+2} + u_{2i}) + u_{2i+3} + u_{2i-1}] -$$

$$- g(u_{2i+1} - v_{2i+1}), \qquad i = 1,\ldots,\frac{M-3}{2},$$

$$\dot{u}_M = 0.$$

We do not give the equations for the v_i-components, because these are now easy to find.

We integrated two systems, viz. system I and II obtained by setting M = 31 and M = 61, respectively. Both systems were integrated over the interval [0,20] by calling M3RK for XE = 10^{-2} (first call), and for XE = 10^{-1},1,5,10,20 (subsequent calls). They were integrated for 3 values of TOL, viz. 10^{-3},10^{-4} and 10^{-5}. The parameter INFO(2) was chosen equal to 2, indicating the second option for the spectral radius (the specification of the remaining input parameters should be clear from the foregoing).

Results of the integrations are listed in tables 5.1 and 5.2. Table 5.1 gives the system I - approximations, and system II - approximations to u(x,t) for the specified number of output times and some grid values x_i. The approximations were rounded to 4 decimal places. Comparing the results for a system for several values of TOL yields an indication about the accuracy of the time integration. Comparing the results for both systems for several values of TOL yields an indication about the accuracy of the space discretization. From this table we thus have an indication about the accuracy of the approximations to u(x,t), i.e. the solution of one of the components of the partial differential equation, at several times and points.

To get some insight in the course of the time integrations, and thus in the behaviour of M3RK, table 5.2 gives for all integrations for each output time the following information: step = the total number of steps, restep = the number of rejected steps, fev = the total number of f(y)-evaluations, sig = the number of f(y)-evaluations needed for the estimation

and control of the spectral radius, sigma = the estimation of the spectral radius used by M3RK. We observe that for both systems no step rejections occurred. Among others, table 5.2 clearly shows the increase of the stepsize during the process. Because of the fact that the problem possesses a steady state solution, such an increase must occur. From this table we can also calculate the average number of function evaluations per step at the given output times. We also see that the average number of function evaluations decreases as TOL becomes smaller. This is due to the fact that for the present problem the stepsize is mostly restricted by accuracy requirements. If this is the case the degree is minimized. As a consequence, a significantly larger number of steps, due to a smaller value value of TOL, not always yields a significantly larger number of function evaluations. This fact is clearly illustrated by table 5.2. It may thus be preferred to choose TOL not too large. Moreover, the various control mechanisms work better for smaller values of TOL.

We conclude this example by noting that all integrations were performed without the necessity of making restarts.

5.2. A NON-LINEAR, TWO-DIMENSIONAL PROBLEM

The second problem we consider is a two-dimensional diffusion problem:

$$
\begin{aligned}
&\frac{\partial u}{\partial t} = \beta(u)\,\Big[\frac{\partial^2 u}{\partial z^2} + \frac{1}{r}\,\frac{\partial}{\partial r}\,\Big(r\frac{\partial u}{\partial r}\Big)\Big] + \gamma(u), \quad 0 \le r \le r_e,\ 0 \le z \le z_e,\\
&u(0,r,z,) = 500, \quad 0 \le r \le r_e,\ 0 \le z \le z_e,\\
&u(t,r,0) = u(t,r,z_e) = 500, \quad 0 \le r \le r_e,\ t > 0,\\
&u_r(t,0,z) = 0,\ u_r(t,r_e,z) = \mu(u(t,r_e,z)),\ 0 \le z \le z_e,\ t > 0,
\end{aligned}
\tag{5.4}
$$

where $r_e = 10^{-4}$, $z_e = 0.15$, and $\beta(u) = \lambda(u)/\rho(u)$, $\gamma(u) = \eta(u)/\rho(u)$, $\mu(u) = \psi(u)/\lambda(u)$. The functions λ, ρ, η, and ψ are defined by

	t \ x	system I						system II					
		0	0.2	0.4	0.6	0.8	0.9	0	0.2	0.4	0.6	0.8	0.9
TOL = 10^{-3}	10^{-2}	.5283	.6879	.6879	.6879	.6879	.6883	.5498	.6879	.6879	.6879	.6879	.6883
	10^{-1}	.2193	.4742	.5105	.5124	.5242	.5773	.2219	.4743	.5105	.5123	.5225	.5717
	1	.0422	.1988	.3676	.5122	.6516	.7377	.0424	.1981	.3664	.5102	.6489	.7345
	5	.0330	.1636	.3225	.4810	.6396	.7318	.0332	.1626	.3207	.4784	.6364	.7283
	10	.0327	.1623	.3204	.4785	.6375	.7301	.0330	.1617	.3190	.4764	.6347	.7269
	20	.0326	.1623	.3203	.4785	.6375	.7300	.0329	.1617	.3190	.4764	.6347	.7269
TOL = 10^{-4}	10^{-2}	.5271	.6870	.6870	.6870	.6870	.6874	.5488	.6870	.6870	.6870	.6870	.6874
	10^{-1}	.2184	.4738	.5099	.5118	.5236	.5767	.2209	.4739	.5099	.5118	.5219	.5711
	1	.0419	.1981	.3671	.5123	.6521	.7381	.0422	.1977	.3660	.5103	.6492	.7348
	5	.0330	.1637	.3226	.4811	.6398	.7318	.0332	.1629	.3210	.4788	.6368	.7285
	10	.0327	.1624	.3204	.4786	.6375	.7301	.0329	.1617	.3190	.4765	.6348	.7270
	20	.0327	.1623	.3204	.4785	.6375	.7301	.0329	.1617	.3190	.4764	.6347	.7269
TOL = 10^{-5}	10^{-2}	.5268	.6867	.6867	.6867	.6867	.6871	.5485	.6867	.6867	.6867	.6867	.6871
	10^{-1}	.2181	.4737	.5098	.5117	.5234	.5766	.2206	.4738	.5098	.5116	.5217	.5709
	1	.0419	.1979	.3670	.5124	.6523	.7383	.0422	.1975	.3659	.5103	.6493	.7349
	5	.0330	.1637	.3226	.4811	.6397	.7318	.0332	.1630	.3213	.4791	.6370	.7287
	10	.0328	.1624	.3205	.4786	.6376	.7302	.0329	.1617	.3190	.4765	.6348	.7270
	20	.0327	.1623	.3204	.4785	.6375	.7301	.0329	.1617	.3190	.4764	.6347	.7269

Table 5.1 Approximations to u(t,x) of (5.1) at several times and points.

		system I					system II				
	t	step	re-step	fev	sig	sigma	step	re-step	fev	sig	sigma
TOL = 10^{-3}	0	0	0	21	21	4462.2	0	0	21	21	6871.4
	10^{-2}	38	0	127	40	1290.9	38	0	127	40	4352.3
	10^{-1}	58	0	183	45	1290.9	58	0	203	45	4352.3
	1	79	0	307	50	1290.9	80	0	415	50	4352.3
	5	98	0	516	50	1290.9	116	0	832	55	4352.3
	10	113	0	669	55	1290.9	145	0	1185	60	4352.3
	20	149	0	1068	60	1290.9	204	0	1908	75	4352.3
TOL = 10^{-4}	0	0	0	21	21	4462.2	0	0	21	21	6871.4
	10^{-2}	66	0	215	65	1185.0	66	0	194	44	4892.5
	10^{-1}	103	0	308	75	1185.0	104	0	318	54	4892.5
	1	142	0	469	80	1185.0	143	0	614	59	4892.5
	5	171	0	725	85	1185.0	210	0	1266	74	4892.5
	10	195	0	949	90	1185.0	244	0	1675	79	4892.5
	20	213	0	1165	95	1185.0	310	0	2482	94	4892.5
TOL = 10^{-5}	0	0	0	21	21	4462.2	0	0	21	21	6871.4
	10^{-2}	122	0	375	99	1304.2	122	0	344	68	4704.6
	10^{-1}	195	0	554	114	1304.2	196	0	541	83	4704.6
	1	273	0	819	129	1304.2	273	0	972	98	4704.6
	5	323	0	1189	139	1304.2	377	0	1931	123	4704.6
	10	358	0	1508	149	1304.2	445	0	2556	133	4704.6
	20	383	0	1775	154	1304.2	509	0	3339	148	4704.6

Table 5.2 Information about M3RK concerning (5.3).

$$\lambda(u) = 418.4 * \{0.1287496_{10}{-}13u^4 - 0.116873_{10}{-}9u^3 +$$
$$+ 0.384891_{10}{-}6u^2 - 0.569812_{10}{-}3u + 0.57185303\},$$
$$\rho(u) = 19300 * \{0.14644_{10}3 + 0.18513_{10}{-}1 * (u{-}0.104_{10}4)\},$$
$$\eta(u) = 7.1486^2 * \pi^{-2} * 10^{10} * \{-0.378808_{10}{-}1 + 0.267688_{10}{-}3u +$$
$$+ 0.1724588_{10}{-}7u^2\},$$
$$\psi(u) = 0.56696_{10}{-}7u^4 * \{-0.270491_{10}{-}17u^5 + 0.3438691_{10}{-}13u^4 -$$
$$- 0.163905_{10}{-}9u^3 + 0.3305647_{10}{-}6u^2 - 0.1194_{10}{-}3u +$$
$$+ 0.2667112_{10}{-}1\}.$$

(5.5)

Problem (5.4) describes the temperature in a filament and was communicated to us by POLAK [7], who is gratefully acknowledged for his cooperation.

For the semi-discretization of (5.4) we have applied a Galerkin method based on piecewise bilinear functions. We have made use of an implementation, written by BAKKER [2], yielding a purely explicit initial value problem. Again we shall confine ourselves to the definition of the resulting system of ordinary differential equations.

Let $\{r_i | r_1 = 0,\ r_i < r_{i+1},\ i = 1,\ldots,M-1,\ r_M = r_e\}$ and $\{z_j | z_1 = 0,\ z_j < z_{j+1},\ j = 1,\ldots,N-1,\ z_N = z_e\}$ be partitions of the r-axis and z-axis, respectively. Let $p_i(r)$, $i = 1,\ldots,M$, and $q_j(z)$, $j = 1,\ldots,N$, be piecewise linear functions, i.e. $p_i(r_k) = \delta_{ik}$ and $q_j(z_1) = \delta_{j1}$, where δ denotes the Kronecker symbol. Further, let $u_{ij}(t) \approx u(t,r_i,z_j)$. The resulting system of ordinary differential equations is then defined by

$$\dot{u}_{ij} = 0, \qquad i = 1,\ldots,M,\ j = 1,N,$$
$$\dot{u}_{ij} = v_i^{-1} w_j^{-1} \beta(u_{ij}) \{r_e w_j \mu(u_{Mj}) \delta_{iM} - a_{ij}u_{ij} - a_{ij-1}u_{ij-1} -$$
$$- a_{ij+1}u_{ij+1} - a_{i-1j}u_{i-1j} - a_{i+1j}u_{i+1j}\} + \gamma(u_{ij}),$$
$$i = 1,\ldots,M, \quad j = 2,\ldots,N-1,$$

(5.6)

where

$$v_i = \int_0^{r_e} rp_i(r)dr,\ w_j = \int_0^{z_e} q_j(z)dz,$$

(5.7)

$$a_{kl} = \int_0^{r_e}\int_0^{z_e} r[\dot{p}_i(r)\dot{p}_k(r)q_j(z)q_l(z) + p_i(r)p_k(r)\ \dot{q}_j(z)\dot{q}_l(z)]\ dr\ dz.$$

Again M3RK was applied to two systems. For both systems $r_i = (i-1)r_e/(M-1)$, $i = 1,\ldots,M$, and $z_j = 0.06*(j-1)/(N-1)$, $j = 1,\ldots,(N-1)/4$; $z_j = z_{j-1} + 0.24/(N-1)$, $j = (N+3)/4,\ldots,(3N+1)/4$; $z_j = z_{j-1} + 0.06/(N-1)$, $j = (3N+5)/4,\ldots,N$. The choice M=5, N=21 and M=5, N=41 yields system I and system II, respectively. Because of the physical nature of the problem it is not necessary to choose M > 5. A finer grid at the endpoints of the filament is necessary in order to deal with boundary layers. It should be observed that, because of two reasons, the resulting systems are difficult problems for our explicit integrator. Firstly, the systems are strongly non-linear. Secondly,in spite of the relatively small number of gridpoints, we have to deal with a large spectral radius because of the very small size of the r-interval.

Both systems were integrated over the interval [0,1] by calling M3RK for XE = 0.1 (first call), and for XE = 0.2,0.4,0.6,0.8,0.9 and 1 (subsequent calls). They were integrated for 3 values of TOL, viz. 10^{-3},10^{-4} and 10^{-5}. Again, the parameter INFO(2) was chosen equal to 2.

Some results of the integrations are listed in tables 5.3 and 5.4. Table 5.3 gives the system I-approximations and system II-approximations to $u_{1j}(t)$ at the specified output times for some values of $z_j \in [0,z_e/2]$. Results for $z \in [z_e/2,z_e]$ are omitted, as the solution is approximately symmetric with respect to $z_e/2$. With respect to comparing and interpreting results of table 5.3, we refer to the remarks made at the preceding example.

Table 5.4 yields the same type of information as table 5.2.The rejections clearly indicate that the guidance of the process is better for smaller values of TOL. Because of this, and because of the minimizing property of the degree, we again conclude that it may be preferred to chose TOL not too large. This conclusion particularly applies when integrating stongly non-linear problems.

We conclude this section by observing that all integrations were performed without the necessity of making restarts.

		system I					system II				
	t \ z	0.003	0.006	0.009	0.015	0.075	0.003	0.006	0.009	0.015	0.075
TOL = 10^{-3}	0.1	700.8	748.9	757.1	758.3	758.4	706.0	751.4	757.1	757.6	757.5
	0.2	951.3	1123.6	1172.4	1185.2	1185.2	955.7	1129.9	1174.5	1183.8	1183.9
	0.4	1746.2	2316.1	2514.2	2593.4	2596.3	1736.1	2322.0	2520.3	2591.1	2593.5
	0.6	2551.3	3154.0	3284.7	3323.9	3325.4	2508.5	3156.7	3288.8	3324.0	3325.5
	0.8	2816.7	3304.4	3382.0	3397.2	3397.6	2755.4	3308.0	3384.3	3396.9	3397.2
	0.9	2847.8	3317.5	3388.2	3400.7	3400.9	2785.6	3321.9	3390.8	3400.6	3400.8
	1.0	2858.8	3321.9	3390.1	3401.5	3401.6	2797.2	3326.7	3392.7	3401.5	3401.6
TOL = 10^{-4}	0.1	700.8	748.9	757.1	758.3	758.4	706.1	752.0	757.8	758.3	758.3
	0.2	952.1	1123.9	1172.4	1185.6	1185.8	956.3	1131.0	1176.0	1185.6	1185.6
	0.4	1746.5	2316.9	2514.8	2594.0	2596.9	1737.4	2323.8	2522.4	2593.6	2595.9
	0.6	2552.1	3153.9	3285.3	3324.4	3325.9	2509.6	3157.7	3289.6	3324.7	3326.2
	0.8	2815.3	3303.7	3381.6	3397.0	3397.3	2756.0	3308.8	3384.8	3397.2	3397.6
	0.9	2847.2	3317.3	3388.1	3400.6	3400.8	2786.4	3322.2	3391.0	3400.7	3400.9
	1.0	2858.6	3321.9	3390.1	3401.5	3401.6	2797.4	3326.8	3392.8	3401.5	3401.6
TOL = 10^{-5}	0.1	701.0	749.5	758.0	759.3	759.3	706.3	752.6	758.7	759.3	759.3
	0.2	952.8	1125.3	1174.2	1187.7	1187.9	957.0	1132.5	1178.0	1187.8	1187.9
	0.4	1748.9	2319.8	2517.7	2597.1	2600.0	1740.0	2327.4	2526.2	2597.6	2600.0
	0.6	2552.1	3153.3	3284.7	3323.7	3325.2	2509.4	3156.9	3288.6	3323.7	3325.2
	0.8	2815.3	3303.5	3381.5	3396.9	3397.2	2755.6	3308.1	3384.3	3396.9	3397.2
	0.9	2846.7	3317.0	3388.0	3400.6	3400.8	2785.7	3321.8	3390.8	3400.6	3400.8
	1.0	2858.3	3321.7	3390.0	3401.5	3401.6	2797.1	3326.7	3392.7	3401.5	3401.6

Table 5.3 Approximations to u(t,0,z) of (5.6) at several times and points.

	system I						system II				
	t	step	re-step	fev	sig	sigma	step	re-step	fev	sig	sigma
TOL = 10^{-3}	0	0	0	11	11	433026.3	0	0	11	11	433627.5
	0.1	146	5	1595	92	361136.0	114	1	1241	73	387114.9
	0.2	275	8	3043	191	314911.0	245	6	2759	206	353507.5
	0.4	483	14	5418	377	250766.5	460	10	5172	340	291676.3
	0.6	666	21	7466	543	209500.5	652	15	7306	468	253444.2
	0.8	767	22	8593	593	195533.7	865	24	9578	615	243312.1
	0.9	803	22	8992	598	195533.7	904	24	10042	625	243312.1
	1.0	830	22	9321	603	195533.7	951	25	10600	648	240319.6
TOL = 10^{-4}	0	0	0	11	11	433026.3	0	0	11	11	433627.5
	0.1	176	0	1548	53	381046.8	176	0	1541	54	383557.4
	0.2	327	0	3059	142	315213.0	336	0	3108	130	353077.8
	0.4	573	1	5395	238	260536.2	617	3	5714	271	286028.2
	0.6	776	1	7402	311	210908.2	854	5	7915	367	257379.4
	0.8	984	9	9437	560	193543.2	1003	6	9373	410	244102.1
	0.9	1026	9	9877	565	193543.2	1038	6	9803	420	244102.1
	1.0	1053	9	10206	570	193543.2	1098	7	10413	443	240106.2
TOL = 10^{-5}	0	0	0	11	11	433026.3	0	0	11	11	433627.5
	0.1	237	0	2131	63	388688.4	237	0	2131	63	390626.6
	0.2	425	0	3902	140	327591.8	440	0	4049	152	357651.3
	0.4	739	0	6858	267	249339.1	792	0	7291	257	306499.2
	0.6	996	0	9237	332	233658.6	1088	0	10047	347	259726.0
	0.8	1231	0	11294	382	233658.6	1337	0	12198	397	259726.0
	0.9	1299	0	11901	392	233658.6	1403	0	12831	412	259726.0
	1.0	1340	0	12372	402	233658.6	1444	0	13315	417	259726.0

Table 5.4 Information about M3RK concerning (5.6)

6. CONCLUDING REMARKS

The purpose of this paper was to develop a robust and efficient time integrator, being easy to apply to a wide class of linear, and in particular non-linear, semi-discretized parabolic problems in one and more dimensions. Although it is not clear that, e.g. for a two-dimensional non-linear problem, our method needs less computer time than an unconditionally stable method (such as an implicit method or an ADI method), the easy applicability of the subroutine, when used in conjunction with semi-discretization, reduces the human effort needed to solve a problem under consideration. For example, the subroutine is easy to use with a software interface performing the semi-discretization (see e.g. SINCOVEC & MADSEN [11] or BAKKER [2]).

It is of interest to observe that until now stabilized, explicit methods, as well as their implementations, did not receive very much attention in literature. As a consequence, it is most likely that the present implementation and its underlying algorithm can be improved significantly.

Acknowledgement

The author wishes to thank Professor P.J. van der Houwen and Professor T.J. Dekker of the University of Amsterdam for their constructive comments and careful reading of the manuscript. The author is also grateful to Mr. M. Bakker for his assistance in programming the Galerkin discretizations, and to Mr. B. Sommeijer for his assistance in testing the program.

REFERENCES

[1] BAKKER, M., *Software for semi-discretization of time dependent partial differential equations in one space variable*, Report NW, Mathematisch Centrum, Amsterdam, (to appear).

[2] BAKKER, M., *Unpublished manuscript*, Mathematisch Centrum, Amsterdam.

[3] BEENTJES, P.A., *NUMAL, a library of ALGOL 60 procedures in numerical mathematics*, section 5.2.1.1.1.1., procedure ARK, Mathematisch Centrum, Amsterdam, 1974.

[4] DOUGLAS, J. & T. DUPONT, *Galerkin methods for parabolic equations*, SIAM J. Numer. Anal. 7, pp. 575-626, 1970.

[5] GEAR, C.W. *Numerical initial value problems in ordinary differential equations*, Prentice Hall, Englewood Cliffs, New Yersey, 1971.

[6] LINDBERG, B., *IMPEX - A program package for solution of systems of stiff differential equations*, Report NA 72.50, The Royal Institute of Technology, Stockholm, 1972.

[7] POLAK, S.J., *Private communication*, ISA, Gebouw VM, Philips, Eindhoven, The Netherlands.

[8] RICHTMEYER, R.D. & K.W. MORTON, *Difference methodes for initial value problems*, Interscience, New York, 1967.

[9] SNYDER, B.G., *The PFORT verifier, Software Practice and Experience*, 4, pp. 359-378, 1974.

[10] SHAMPINE, L. & M.K. GORDON, *Computer solution of ordinary differential equations*, The initial value problem, W.H. Freeman and Co., San Francisco, 1975.

[11] SINCOVEC, R.F. & N.K. MADSEN, *Software for non-linear partial differential equations*, ACM Transactions on mathematical software 1, pp. 232-260, 1975.

[12] TE RIELE, H.J.J., (ed.), *Colloquium Numerieke Programmatuur* (Dutch), MC syllabus, Mathematisch Centrum, Amsterdam, (to appear).

[13] VAN DER HOUWEN, P.J., *Construction of integration formulas for initial*

value problems, North-Holland Publishing Company, Amsterdam, 1976.

[14] VERWER, J.G., *A class of stabilized three-step Runge-Kutta methods for the numerical integration of parabolic equations*, Journal of Computational and Applied Mathematics, (to appear).

[15] VERWER, J.G., *Multipoint multistep Runge-Kutta methods I: On a class of two-step methods for parabolic equations*, Report NW30/76, Mathematisch Centrum, Amsterdam, 1976.

[16] VERWER, J.G., *Multipoint multistep Runge-Kutta methods II: The construction of a class of stabilized three-step methods for parabolic equations*, Report NW31/76, Mathematisch Centrum, Amsterdam, 1976.

```
      SUBROUTINE M3RK(X,XE,N,H,HMIN,SIGMA,TOL,F,Y,
     +                Y1,Y2,YXE,DY,DY1,IFLAG,INFO)
C***********************************************************************
C* ABSTRACT                                                            *
C***********************************************************************
C* M3RK IS DESIGNED TO SOLVE INITIAL VALUE PROBLEMS FOR SYSTEMS OF     *
C* ORDINARY DIFFERENTIAL EQUATIONS OF THE FORM                         *
C*                    DY/DX=F(Y(1),...,Y(N)),                          *
C*                    Y(I) GIVEN AT X,                                 *
C* WHICH ORIGINATE FROM SEMI-DISCRETIZATION OF INITIAL-BOUNDARY VALUE  *
C* PROBLEMS FOR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS. MR3K IS      *
C* BASED ON STABILIZED,EXPLICIT THREE-STEP RUNGE-KUTTA FORMULAS OF     *
C* ORDER ONE AND TWO,OF WHICH THE DEGREE CAN VARY BETWEEN 2 AND 12.    *
C*                                                                     *
C* M3RK NEEDS 6 ARRAYS OF LENGTH N,WHICH ALL APPEAR IN THE CALL LIST.  *
C* THE CODE INTEGRATES FROM X TO XE.ON NORMAL RETURN THE PARAMETERS IN *
C* THE CALL LIST ARE READY FOR CONTINUING THE INTEGRATION.TO CONTINUE  *
C* THE INTEGRATION,THE USER NEEDS ONLY TO REDEFINE THE OUTPUT POINT XE *
C* AND CALL AGAIN.                                                     *
C*                                                                     *
C* M3RK CALLS 11 SUBROUTINES WHICH HAVE BEEN WRITTEN TO STRUCTURE THE  *
C* PROGRAM.THESE SUBROUTINES ARE:                                      *
C* HSTART - HSTART COMPUTES THE INITIAL STEPLENGTH                     *
C* PARAM  - PARAM COMPUTES PARAMETERS OF THE VARIOUS IMPLEMENTED       *
C*          SCHEMES FROM THE COEFFICIENTS OF THE STABILITY POLYNOMIALS *
C* POWERM - POWERM ESTIMATES THE SPECTRAL RADIUS OF THE JACOBIAN OF F  *
C* MAXDEG - MAXDEG COMPUTES THE MAXIMAL DEGREE OF THE FORMULAS,        *
C*          WHICH IS ALLOWED WITH RESPECT TO INTERNAL STABILITY        *
C* MINDEG - MINDEG COMPUTES THE MINIMAL DEGREE OF THE FORMULAS,        *
C*          WHICH IS ALLOWED WITH RESPECT TO ABSOLUTE STABILITY        *
C* STEP   - STEP CONTAINS THE ACTUAL INTEGRATOR                        *
C* ESTIMA - ESTIMA COMPUTES A LOCAL ERROR BOUND AND ESTIMATES A LOCAL  *
C*          ERROR                                                      *
C* NEWH   - NEWH DELIVERS A NEW STEPLENGTH                             *
C* INTER1 - INTER1 PERFORMS INTERPOLATION AFTER A CHANGE OF THE        *
C*          STEPLENGTH                                                 *
C* INTER2 - INTER2 INTERPOLATES THE SOLUTION AT THE OUTPUT POINT XE    *
C* SHIFT  - SHIFT SHIFTS THE DATA FOR A NEXT STEP                      *
C*                                                                     *
C* THESE SUBROUTINES ARE COMPLETELY LOCAL,I.E.THE INFORMATION THEY     *
C* NEED IS PASSED THROUGH THE PARAMETER LISTS.THE WHOLE PACKAGE HAS    *
C* BEEN TESTED ON A CDC CYBER 73-28 USING AN ARITHMETIC PRECISION OF   *
C* 14 DIGITS.THE CODE POWERM USES THE CDC SYSTEM SUBPROGRAMS RANSET    *
C* AND RANF CONSTITUTING A RANDOM GENERATOR.RANSET AND RANF MUST BE    *
C* REPLACED WHEN USING THE PROGRAM ON ANOTHER COMPUTER.                *
C* THE CODES CALLED BY M3RK USE NO MACHINE DEPENDENT CONSTANTS.        *
C* M3RK USES ONE MACHINE DEPENDENT CONSTANT,NAMELY THE ARITHMETIC      *
C* PRECISION OF THE COMPUTER.IN THE PROGRAM THE INTERNAL VARIABLE APR, *
C* WHICH REPRESENTS THE ARITHMETIC PRECISION,EQUALS 1.0E-14. APR MUST  *
C* BE CHANGED ACCORDINGLY WHEN USING THE PROGRAM ON ANOTHER COMPUTER.  *
C*                                                                     *
C* THE WHOLE PROGRAM PACKAGE IS ACCEPTED BY THE PFORT VERIFIER.THE     *
C* PFORT VERIFIER IS A PROGRAM WHICH CHECKS A FORTRAN PROGRAM,I.E. A    *
C* MAIN PROGRAM AND SUBPROGRAMS,FOR ADHERENCE TO PFORT,A PORTABLE      *
C* SUBSET OF AMERICAN NATIONAL STANDARD FORTRAN(SEE[1]).THE WHOLE      *
C* PROGRAM PACKAGE IS COMPLETELY EXPLAINED AND DESCRIBED IN [2].       *
C*                                                                     *
```

```
C* [1] RYDER,B.G.,THE PFORT VERIFIER,SOFTWARE PRACTICE AND EXPERIENCE, *
C*     VOL.4,PP.359-378,1974.
C* [2] VERWER,J.G.,AN IMPLEMENTATION OF A CLASS OF STABILIZED,EXPLICIT *
C*     METHODS FOR THE TIME INTEGRATION OF PARABOLIC EQUATIONS,(TO     *
C*     APPEAR).                                                        *
C***********************************************************************
C* MEANING OF THE PARAMETERS                                           *
C***********************************************************************
C* X      - VARIABLE    : INDEPENDENT VARIABLE                         *
C* XE     - EXPRESSION  : OUTPUT POINT AT WHICH SOLUTION IS DESIRED    *
C* N      - EXPRESSION  : NUMBER OF EQUATIONS                          *
C* H      - VARIABLE    : STEPLENGTH                                   *
C* HMIN   - VARIABLE    : MINIMAL STEPLENGTH                           *
C* SIGMA  - ARRAY       : AN ARRAY OF LENGTH 2 CONTAINING ESTIMATES    *
C*                        OF THE SPECTRAL RADIUS OF THE JACOBIAN OF F  *
C* TOL    - EXPRESSION  : LOCAL ERROR TOLERANCE                        *
C* F      - SUBROUTINE  : DERIVATIVE                                   *
C* Y      - ARRAY       : SOLUTION VECTOR AT X,INPUT AND WORK ARRAY    *
C* Y1     - ARRAY       : SOLUTION VECTOR AT X-H,WORK ARRAY            *
C* Y2     - ARRAY       : SOLUTION VECTOR AT X-2H,WORK ARRAY           *
C* YXE    - ARRAY       : SOLUTION VECTOR AT XE,OUTPUT AND WORK ARRAY  *
C* DY     - ARRAY       : DERIVATIVE VECTOR AT X,WORK ARRAY            *
C* DY1    - ARRAY       : DERIVATIVE VECTOR AT X-H,WORK ARRAY          *
C* IFLAG  - VARIABLE    : ERROR FLAG                                   *
C* INFO   - ARRAY       : INTEGER ARRAY OF LENGTH 15.INFO IS USED TO   *
C*                        PASS INFORMATION TO INITIALIZE THE CODE,TO   *
C*                        PASS INFORMATION BETWEEN THE MAIN PROGRAM AND*
C*                        THE SUBPROGRAMS,TO PASS INFORMATION TO THE  *
C*                        USER ABOUT THE STATUS OF THE INTEGRATION,AND *
C*                        TO RETAIN INFORMATION FOR SUBSEQUENT CALLS.  *
C***********************************************************************
C* FIRST CALL TO M3RK                                                  *
C***********************************************************************
C* THE USER MUST PROVIDE STORAGE IN HIS CALLING PROGRAM FOR THE ARRAYS *
C* IN THE CALL LIST.HE HAS TO SUPPLY THE SUBROUTINE F(N,Y) FOR         *
C* EVALUATING THE DERIVATIVES DY(I)/DT,I=1,...,N,WHICH MUST BE OVER-   *
C* WRITTEN ON Y(I).FOR THE SPECTRAL RADIUS THERE EXIST THREE OPTIONS    *
C* WHICH MUST BE SELECTED WITH INFO(2).IN INFO(3) THE USER MUST GIVE    *
C* A MAXIMUM FOR THE NUMBER OF EVALUATIONS OF F(Y) TO BE SPENT(ON RE-   *
C* TURN IT MAY OCCUR THAT INFO(3) IS EXCEEDED WITH ABOUT 50)           *
C************************                                             *
C* INPUT PARAMETERS                                                    *
C************************                                             *
C* X      - INITIAL VALUE OF THE INDEPENDENT VARIABLE                  *
C* XE     - OUTPUT POINT AT WHICH SOLUTION IS DESIRED                  *
C* N      - NUMBER OF EQUATIONS                                        *
C* SIGMA  - (1)= AN UPPER ESTIMATION OF THE SPECTRAL RADIUS OF THE JA- *
C*              COBIAN OF F IN CASE OF OPTION 1.FOR OPTION 2 AND 3 NO  *
C*              INITIALIZATION IS REQUIRED                             *
C* TOL    - LOCAL ERROR TOLERANCE                                      *
C* Y      - VECTOR OF INITIAL VALUES OF THE DEPENDENT VARIABLES        *
C* INFO   - (1)= 0 TO INDICATE FIRST CALL                              *
C*          (2)= 1 TO INDICATE OPTION 1 FOR THE SPECTRAL RADIUS,I.E.   *
C*               THE USER MUST INITIALIZE SIGMA(1)                     *
C*             = 2 TO INDICATE OPTION 2 FOR THE SPECTRAL RADIUS,I.E.   *
C*               THE CODE INITIALIZES SIGMA(1) AND CONTROLS SIGMA(1)   *
C*             = 3 TO INDICATE OPTION 3 FOR THE SPECTRAL RADIUS,I.E.   *
```

```
C*                  THE CODE ONLY INITIALIZES SIGMA(1) AT THE FIRST CALL *
C*          (3)= MAXIMUM NUMBER OF F(Y) EVALUATIONS TO BE SPENT          *
C*************************                                              *
C* OUTPUT PARAMETERS                                                    *
C*************************                                              *
C* X      - LAST POINT REACHED IN INTEGRATION.NORMALLY X IS SLIGHTLY     *
C*          BEYOND XE                                                   *
C* H      - INITIAL STEPLENGTH FOR SUBSEQUENT CALL                      *
C* HMIN   - MINIMAL STEPLENGTH USED BY M3RK                             *
C* SIGMA  - (1)= UPPER ESTIMATION OF THE SPECTRAL RADIUS INITIALIZED BY *
C*               THE USER OR BY POWERM                                  *
C*        - (2)= INACCURATE ESTIMATION OF THE SPECTRAL RADIUS USED FOR  *
C*               ITS CONTROL                                            *
C* Y      - SOLUTION VECTOR AT X                                        *
C* Y1     - SOLUTION VECTOR AT X-H                                      *
C* Y2     - SOLUTION VECTOR AT X-2H                                     *
C* YXE    - SOLUTION VECTOR AT OUTPUT POINT XE                          *
C* DY     - DERIVATIVE VECTOR AT X                                      *
C* DY1    - DERIVATIVE VECTOR AT X-H                                    *
C* IFLAG  - = 0 NORMAL RETURN,I.E.OUTPUT POINT IS REACHED               *
C*          = 1 OUTPUT POINT IS NOT REACHED.THE MAXIMUM NUMBER OF       *
C*              F(Y)-EVALUATIONS HAS BEEN SPENT.THE PROCESS CAN BE CON- *
C*              TINUED BY INCREASING INFO(3) AND CALLING AGAIN          *
C*          = 2 MAXIMAL DEGREE FALLS OUTSIDE THE RANGE AS TOL/APR IS    *
C*              TOO SMALL.THE PROCESS IS NOT STARTED                    *
C*          = 3 POWERM FAILED IN THE ESTIMATION OF THE SPECTRAL RADIUS. *
C*              THE PROCESS IS DISCONTINUED                             *
C* INFO   - (1) = 1 TO INDICATE THAT THE NEXT CALL IS A SUBSEQUENT ONE  *
C*          (2) = 1 IN CASE OF OPTION 1 OR 3,ELSE 2                     *
C*          (3) = UNCHANGED                                             *
C*          (4) = TOTAL NUMBER OF INTEGRATION STEPS PERFORMED,I.E.       *
C*                ACCEPTED AND REJECTED ONES                            *
C*          (5) = NUMBER OF REJECTED INTEGRATION STEPS                  *
C*          (6) = NUMBER OF RESTARTS INITIATED BY THE CODE              *
C*          (7) = TOTAL NUMBER OF DERIVATIVE EVALUATIONS                *
C*          (8) = NUMBER OF DERIVATIVE EVALUATIONS USED FOR THE ESTIMA- *
C*                TION AND CONTROL OF THE SPECTRAL RADIUS               *
C*          (9) = CURRENT DEGREE                                        *
C*          (10)= MAXIMAL DEGREE FOR THE FIRST ORDER FORMULAS           *
C*          (11)= MAXIMAL DEGREE FOR THE SECOND ORDER FORMULAS          *
C*          (12)= CURRENT ORDER                                         *
C*          (13)= NUMBER OF STEPS PERFORMED AFTER START OR RESTART      *
C*          (14)= NUMBER OF STEPS PERFORMED AFTER CHANGE OF H OR ORDER  *
C*          (15)= NUMBER OF STEPS PERFORMED AFTER ESTIMATION OF SPEC-   *
C*                TRAL RADIUS                                           *
C*                                                                      *
C* IT IS EMPHASIZED THAT THE OUTPUT LIST GIVEN ABOVE IS NOT STRICTLY    *
C* VALID WHEN ON RETURN IFLAG IS EQUAL TO 2 OR 3.                       *
C* IT IS FURTHER EMPHASIZED THAT ON RETURN THE ARRAY Y ALWAYS CONTAINS  *
C* THE SOLUTION VECTOR AT THE POINT X.THUS WHEN IFLAG=3,THE USER HAS    *
C* THE POSSIBILITY TO RESTART THE PROCESS AT THE POINT X,PROVIDED HE    *
C* IS ABLE TO TAKE ACTION WITH RESPECT TO THE ESTIMATION OF THE SPEC-   *
C* TRAL RADIUS.                                                         *
C**********************************************************************
C* SUBSEQUENT CALLS TO M3RK                                             *
C**********************************************************************
C* ON RETURN OF M3RK ALL PARAMETERS ARE READY FOR CONTINUING THE INTE-  *
```

```
C* GRATION,PROVIDED IFLAG IS NOT EQUAL TO 2 OR 3.IF XE IS REACHED AND  *
C* A NORMAL CONTINUATION IS DESIRED,THE USER NEED ONLY TO DEFINE A NEW *
C* OUTPUT  POINT XE AND CALL AGAIN.IF ON RETURN IFLAG=1 AND THE USER    *
C* WANTS TO CONTINUE,HE ONLY NEEDS TO INCREASE INFO(3) AND CALL AGAIN. *
C* THE PROGRAM IS WRITTEN IN SUCH A WAY THAT THE CHOICE OF OUTPUT       *
C* POINTS  DOES NOT AFFECT THE INTEGRATION PROCESS ITSELF.BETWEEN SUB-  *
C* SEQUENT CALLS THE USER MAY INCREASE TOL.ALL OTHER PARAMETERS MUST    *
C* REMAIN UNCHANGED.THE COUNTERS IN INFO ARE USED ACCUMULATIVELY.       *
C*********************************************************************
C* PROGRAM TEXT                                                       *
C*********************************************************************
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N),DY(N),DY1(N),SIGMA(2),INFO(15)
C************************************                                  *
C* MEANING OF THE INTERNAL VARIABLES                                  *
C************************************                                  *
C* ALFA    - THE FACTOR FOR CHANGING THE STEPLENGTH                   *
C* APR     - THE ARITHMETIC PRECISION                                 *
C* HOLD    - PREVIOUSLY ACCEPTED STEPLENGTH                           *
C* MOLD    - PREVIOUSLY USED DEGREE                                   *
C* REJECT  - NUMBER OF SUCCESSIVE STEP FAILURES                       *
C* B       - NONZERO B-PARAMETERS OF THE SCHEME                       *
C* C       - C-PARAMETERS OF THE SCHEME                               *
C* LA      - LABDA-PARAMETERS OF THE SCHEME                           *
C* HMAX    - MAXIMAL STEPSIZES WITH RESPECT TO ABSOLUTE STABILITY     *
C* EPS     - LOCAL ERROR BOUND                                        *
C* ERROR   - ESTIMATED LOCAL ERROR                                    *
C*********************************************************************
      INTEGER REJECT
      REAL LA
      DIMENSION B(2),C(12),LA(12),HMAX(2)
C*********************************************************************
C* IF ALREADY PAST OUTPUT POINT DURING A SUBSEQUENT CALL,THEN INTER-   *
C* POLATE AND RETURN                                                  *
C*********************************************************************
      IF(INFO(1).EQ.0.OR.X.LT.XE) GOTO 10
      CALL INTER2(N,Y,Y1,Y2,YXE,(X-XE)/H)
      RETURN
C*********************************************************************
C* SET THE ERROR FLAG IFLAG EQUAL TO ZERO AND INITIALIZE APR.DETERMINE *
C* THE MAXIMAL DEGREES WITH RESPECT TO INTERNAL STABILITY.IF NECESSARY *
C* INTERRUPT                                                          *
C*********************************************************************
   10 IFLAG=0
      APR=1.0E-14
      CALL MAXDEG(TOL,APR,IFLAG,INFO)
      IF(IFLAG.NE.2) GOTO 20
      RETURN
C*********************************************************************
C* SET THE CONTROL VARIABLES INFO(9),REJECT AND HOLD FOR A CONTINUING  *
C* CALL,AND INITIALIZE THE ARRAY HMAX                                 *
C*********************************************************************
   20 INFO(9)=0
      IF(INFO(1).EQ.0) GOTO 30
      REJECT=0
      HOLD=H
      HMAX(1)=5.15*FLOAT(INFO(10))*FLOAT(INFO(10))/SIGMA(1)
      HMAX(2)=2.29*FLOAT(INFO(11))*FLOAT(INFO(11))/SIGMA(1)
```

```
      GOTO 80
C*********************************************************************
C* SET REJECT,HOLD AND H FOR THE FIRST CALL. SET THE ELEMENTS INFO(I), *
C* I=4,...,8,13,14,15 EQUAL TO ZERO AND INFO(12) EQUAL TO 2.TO PREPARE *
C* THE FIRST STEP EVALUATE F(Y) AND SUBSTITUTE INTO DY.IF NECESSARY    *
C* ESTIMATE THE SPECTRAL RADIUS AND CHECK FOR A FAILURE OF THE POWER   *
C* METHOD.INITIALIZE ARRAY HMAX AND ESTIMATE THE INITIAL STEPLENGTH    *
C*********************************************************************
   30 INFO(1)=1
      REJECT=0
      DO 40 I=4,8
   40 INFO(I)=0
      DO 50 I=13,15
   50 INFO(I)=0
      INFO(12)=2
      DO 60 I=1,N
      DY(I)=Y(I)
      DY1(I)=0.
      Y1(I)=0.
      Y2(I)=0.
   60 CONTINUE
      CALL F(N,DY)
      INFO(7)=INFO(7)+1
      IF(INFO(2).EQ.1) GOTO 70
      SIGMA(1)=0.
      CALL POWERM(N,Y,Y1,YXE,DY,DY1,F,SIGMA,APR,IFLAG,INFO)
      IF(IFLAG.NE.3) GOTO 70
      RETURN
C
   70 HMAX(1)=5.15*FLOAT(INFO(10))*FLOAT(INFO(10))/SIGMA(1)
      HMAX(2)=2.29*FLOAT(INFO(11))*FLOAT(INFO(11))/SIGMA(1)
      CALL HSTART(N,Y,DY,YXE,F,TOL,APR,SIGMA(1),HMIN,INFO)
      H=HMIN
      HOLD=H
C*********************************************************************
C* DETERMINE THE DEGREE,AND,IF NECESSARY,CALCULATE THE PARAMETERS OF   *
C* THE SCHEME TO BE USED                                               *
C*********************************************************************
   80 MOLD=INFO(9)
      CALL MINDEG(H,SIGMA(1),INFO)
      IF(INFO(9).EQ.MOLD) GOTO 90
      CALL PARAM(C,LA,B,INFO)
C*********************************************************************
C* CHECK IF THE MAXIMUM NUMBER OF EVALUATIONS IS REACHED.UPDATE HMIN   *
C* AND CALCULATE A SOLUTION AT X+H                                     *
C*********************************************************************
   90 IF(INFO(7).GE.INFO(3)) IFLAG=1
      IF(IFLAG.NE.1) GOTO 100
      RETURN
C
  100 IF(H.LT.HMIN) HMIN=H
      CALL STEP(N,Y,Y1,Y2,YXE,DY,DY1,H,F,C,LA,B,INFO)
      INFO(13)=INFO(13)+1
      INFO(4)=INFO(4)+1
C*********************************************************************
C* IF THE PROCESS IS IN THE START PHASE,SHIFT THE DATA.CHECK FOR THE    *
C* CONTINUATION WITH A THREE-STEP SCHEME.IF THE OUTPUT POINT IS PASSED *
```

```
C* INTERPOLATE AND RETURN                                              *
C**********************************************************************
      IF(INFO(13).GE.3) GOTO 110
      CALL SHIFT(N,Y,Y1,Y2,YXE,DY,DY1,X,HOLD,F,INFO)
      INFO(14)=INFO(14)+1
      INFO(15)=INFO(15)+1
      IF(INFO(13).EQ.1) GOTO 90
      INFO(9)=0
      IF(X.LT.XE) GOTO 80
      CALL INTER2(N,Y,Y1,Y2,YXE,(X-XE)/HOLD)
      RETURN
C**********************************************************************
C* CALCULATE THE LOCAL ERRORBOUND AND ESTIMATE THE LOCAL ERROR.        *
C**********************************************************************
  110 CALL ESTIMA(N,Y,Y1,Y2,YXE,TOL,EPS,ERROR,INFO)
C**********************************************************************
C* IF THE ERROR IS TOO LARGE FOR THE THIRD STEP AFTER START,THEN RE-   *
C* START AT INITIAL POINT WITH H=H/10                                  *
C**********************************************************************
      IF(EPS.GE.ERROR.OR.INFO(13).NE.3) GOTO 130
      INFO(6)=INFO(6)+1
      INFO(5)=INFO(5)+3
      INFO(9)=0
      INFO(13)=0
      INFO(14)=0
      INFO(15)=0
      X=X-2.*H
      H=H/10.
      DO 120 I=1,N
      Y(I)=Y2(I)
      DY(I)=Y(I)
  120 CONTINUE
      CALL F(N,DY)
      INFO(7)=INFO(7)+1
      GOTO 80
C**********************************************************************
C* IF STEP FAILED,CHECK FOR A REESTIMATION OF THE SPECTRAL RADIUS.IF   *
C* NECESSARY,CHECK FOR A FAILURE OF POWERM AND UPDATE HMAX             *
C**********************************************************************
  130 IF(INFO(2).EQ.1.OR.INFO(15).EQ.0.OR.EPS.GE.ERROR) GOTO 150
      SIGMA(1)=0.
      CALL POWERM(N,Y,Y1,YXE,DY,DY1,F,SIGMA,APR,IFLAG,INFO)
      IF(IFLAG.NE.3) GOTO 140
      RETURN
C
  140 HMAX(1)=5.15*FLOAT(INFO(10))*FLOAT(INFO(10))/SIGMA(1)
      HMAX(2)=2.29*FLOAT(INFO(11))*FLOAT(INFO(11))/SIGMA(1)
C**********************************************************************
C* CALCULATE A NEW STEPLENGTH                                          *
C**********************************************************************
  150 HOLD=H
      CALL NEWH(EPS/ERROR,HOLD,H,ALFA,HMAX,INFO)
C**********************************************************************
C* IF THE ORDER EQUALS 1 AND THE STEPLENGTH IS SMALLER THAN THE MAXI-  *
C* MAL STEPLENGTH FOR ORDER 2 RESET THE ORDER                          *
C**********************************************************************
      IF(INFO(12).EQ.1.AND.H.LT.HMAX(2)) INFO(9)=0
```

```
      IF(INFO(9).EQ.0) INFO(12)=2
C*****************************************************************
C* IF STEP FAILED REJECT THE INTEGRATION STEP.CHECK FOR THREE SUCCES-  *
C* SIVE FAILURES,AND,IF NECESSARY,INTERPOLATE FOR THE NEW STEPLENGTH   *
C* ************************************************************
      IF(EPS.GE.ERROR) GOTO 170
      REJECT=REJECT+1
      INFO(5)=INFO(5)+1
      IF(REJECT.EQ.3) GOTO 160
      CALL INTER1(N,Y,Y1,Y2,DY1,F,ALFA,INFO)
      GOTO 80
C*****************************************************************
C* RESET REJECT,INFO(I),I=6,9,12,13,14 AND THE STEPLENGTH FOR A        *
C* RESTART                                                             *
C*****************************************************************
  160 REJECT=0
      INFO(6)=INFO(6)+1
      INFO(9)=0
      INFO(12)=2
      INFO(13)=0
      INFO(14)=0
      CALL HSTART(N,Y,DY,YXE,F,TOL,APR,SIGMA(1),H,INFO)
      HOLD=H
      GOTO 80
C*****************************************************************
C* FIND OUT IF THE ORDER SHOULD CHANGE FROM 2 TO 1.ON EXIT OF THIS     *
C* PROGRAM PART H SHOULD BE RESET TO HMAX(2).IF THE ORDER HAS TO BE    *
C* CHANGED FROM 2 TO 1 SET INFO(9)=0                                   *
C*****************************************************************
  170 IF(INFO(14).LT.3.OR.INFO(12).EQ.1) GOTO 180
      IF(HOLD.NE.HMAX(2).OR.H.NE.HMAX(2)) GOTO 180
      INFO(12)=1
      CALL ESTIMA(N,Y,Y1,Y2,YXE,TOL,EPS,ERROR,INFO)
      CALL NEWH(EPS/ERROR,HOLD,H,ALFA,HMAX,INFO)
      IF(ALFA.LE.1.) INFO(12)=2
      H=HMAX(2)
      INFO(14)=-1
      IF(INFO(12).EQ.2) GOTO 180
      INFO(9)=0
C*****************************************************************
C* SHIFT THE DATA                                                      *
C*****************************************************************
  180 CALL SHIFT(N,Y,Y1,Y2,YXE,DY,DY1,X,HOLD,F,INFO)
      REJECT=0
      INFO(14)=INFO(14)+1
      INFO(15)=INFO(15)+1
C*****************************************************************
C* CHECK FOR A REESTIMATION OF THE SPECTRAL RADIUS.IF NECESSARY,CHECK   *
C* FOR A FAILURE OF POWERM AND UPDATE HMAX                             *
C*****************************************************************
      IF(INFO(2).EQ.1.OR.INFO(15).NE.25) GOTO 200
      CALL POWERM(N,Y,Y1,YXE,DY,DY1,F,SIGMA,APR,IFLAG,INFO)
      IF(IFLAG.NE.3) GOTO 190
      RETURN
C
  190 HMAX(1)=5.15*FLOAT(INFO(10))*FLOAT(INFO(10))/SIGMA(1)
      HMAX(2)=2.29*FLOAT(INFO(11))*FLOAT(INFO(11))/SIGMA(1)
```

```
C*********************************************************************
C* IF X IS SMALLER THAN XE CONTINUE THE INTEGRATION.IF NECESSARY,IN-  *
C* TERPOLATE FOR A CHANGE OF H                                        *
C*********************************************************************
  200 IF(X.GE.XE) GOTO 210
      IF(H.NE.HOLD) CALL INTER1(N,Y,Y1,Y2,DY1,F,ALFA,INFO)
      IF(H.NE.HOLD.OR.INFO(9).EQ.0) GOTO 80
      GOTO 90
C*********************************************************************
C* INTERPOLATE AT XE AND,IF NECESSARY,INTERPOLATE FOR A CHANGE OF H   *
C* BEFORE RETURN                                                      *
C*********************************************************************
  210 CALL INTER2(N,Y,Y1,Y2,YXE,(X-XE)/HOLD)
      IF(H.NE.HOLD) CALL INTER1(N,Y,Y1,Y2,DY1,F,ALFA,INFO)
      RETURN
      END



      SUBROUTINE HSTART(N,Y,DY,YXE,F,TOL,APR,SIGMA,H,INFO)
C*********************************************************************
C* SUBROUTINE HSTART CALCULATES THE INITIAL STEPLENGTH                *
C*********************************************************************
      DIMENSION Y(N),DY(N),YXE(N),INFO(15)
      DO 10 I=1,N
      YXE(I)=Y(I)+DY(I)/SIGMA
   10 CONTINUE
      CALL F(N,YXE)
      INFO(7)=INFO(7)+1
      ETAT=0.0
      ETAE=0.0
      DO 20 I=1,N
      ETAT=ETAT+Y(I)*Y(I)
      E=YXE(I)-DY(I)
      ETAE=ETAE+E*E
   20 CONTINUE
      ETAT=TOL+TOL*SQRT(ETAT/FLOAT(N))
      ETAE=SQRT(ETAE/FLOAT(N))/SIGMA+APR
      H=SQRT(ETAT/ETAE)/SIGMA/10.0
C
      RMMAX=INFO(11)
      BETA=(0.03*RMMAX+0.44)*RMMAX*RMMAX
      IF(H.GT.BETA/SIGMA)H=BETA/SIGMA
      RETURN
      END
```

```
      SUBROUTINE PARAM(C,LA,B,INFO)
C*********************************************************************
C* PARAM DETERMINES THE PARAMETERS OF THE INTEGRATION SCHEME.THESE   *
C* PARAMETERS ARE EXPRESSIONS DEPENDING ON THE COEFFICIENTS OF THE   *
C* STABILITY POLYNOMIALS,WHICH ARE STORED IN THE ARRAY D. DURING     *
C* THE START,I.E.INFO(13) IS 0 OR 1,THE PARAMETERS OF A ONE-STEP SCHE- *
C* ME ARE DETERMINED.                                                *
C*********************************************************************
       REAL LA
       DIMENSION C(12),LA(12),B(2),D(440),P(13),S(13),INFO(15)
       INTEGER ORDER
       DATA D(1),D(2),D(3),D(4),D(5),D(6),D(7),D(8),D(9),D(10),
      +D(11),D(12),D(13),D(14),D(15),D(16),D(17),D(18),D(19),D(20),
      +D(21),D(22),D(23),D(24),D(25),D(26),D(27),D(28),D(29),D(30)/
      +.54545454545454E+00,.32974222139503E00.15874540963720E-01,
      +.54545454545454E+00,.32957779372404E+00,.18807742712872E-01,
      +.26931406295668E-03,.54545454545454E+00,.32959633626966E+00,
      +.19843848141588E-01,.38370516974646E-03,.23214008199836E-05,
      +.54545454545454E+00,.32940480780399E+00,.20289622974884E-01,
      +.43922728369494E-03,.38881393659393E-05,.12058633806519E-07,
      +.54545454545454E+00,.32915730747582E+00,.20529934599533E-01,
      +.47014565070180E-03,.48729959290595E-05,.23293337843875E-07,
      +.41768893290961E-10,.54545454545454E+00,.32940290556199E+00,
      +.20695785946957E-01,.48959305208277E-03,.55250561672575E-05/
       DATA D(31),D(32),D(33),D(34),D(35),D(36),D(37),D(38),D(39),
      +D(40),D(41),D(42),D(43),D(44),D(45),D(46),D(47),D(48),D(49),
      +D(50),D(51),D(52),D(53),D(54),D(55),D(56),D(57),D(58),D(59),
      +D(60)/
      +.32042572866540E-07,.92217187087773E-10,.10431596770258E-12,
      +.54545454545454E+00,.32904622047855E+00,.20769127247467E-01,
      +.50144873108237E-03,.59538312498493E-05,.38413639404624E-07,
      +.13735161731261E-09,.25579038177072E-12,.19355325549260E-15,
      +.54545454545454E+00,.32902403825404E+00,.20835887364795E-01,
      +.51019371265378E-03,.62671263848834E-05,.43273294211670E-07,
      +.17557956622083E-09,.41529099936815E-12,.52977760638010E-15,
      +.28162396623902E-18,.54545454545454E+00,.32900701770523E+00,
      +.20847709891773E-01,.51474022625718E-03,.64655444450446E-05,
      +.46694681553560E-07,.20545752812528E-09,.55985683650846E-12/
       DATA D(61),D(62),D(63),D(64),D(65),D(66),D(67),D(68),D(69),
      +D(70),D(71),D(72),D(73),D(74),D(75),D(76),D(77),D(78),D(79),
      +D(80),D(81),D(82),D(83),D(84),D(85),D(86),D(87),D(88),D(89),
      +D(90)/
      +.92238940881933E-15,.84171480799272E-18,.32655765072494E-21,
      +.54545454545454E+00,.32923047518706E+00,.20942105514450E-01,
      +.52155114179973E-03,.66698403229501E-05,.49787290971999E-07,
      +.23175109073032E-09,.69290153772379E-12,.13309584971259E-14,
      +.15876152476718E-17,.10702794409632E-20,.31159779644428E-24,
      +.54545454545454E+00,.32888824622955E+00,.20930564082556E-01,
      +.52398048369937E-03,.67865318951772E-05,.51892263386503E-07,
      +.25160762750795E-09,.80319349035922E-12,.17105594522426E-14,
      +.24063258323529E-17,.21467888957759E-20,.11002739569540E-23,
      +.24672233557657E-27,.45454545454545E+00,.39753050587769E+00/
       DATA D(91),D(92),D(93),D(94),D(95),D(96),D(97),D(98),D(99),
      +D(100),D(101),D(102),D(103),D(104),D(105),D(106),D(107),
      +D(108),D(109),D(110),D(111),D(112),D(113),D(114),D(115),
      +D(116),D(117),D(118),D(119),D(120)/
      +.19106034236683E-01,.45454545454545E+00,.39769493354868E+00,
```

```
+.22675100922608E-01,.32438614977749E-03,.45454545454545E+00,
+.39767639100306E+00,.23921246939481E-01,.46221334545758E-03,
+.27945492539796E-05,.45454545454545E+00,.39786791946874E+00,
+.24509754044867E-01,.53071848010798E-03,.47002589400275E-05,
+.14585303356181E-07,.45454545454545E+00,.39811541979691E+00,
+.24864589588956E-01,.56998860293388E-03,.59138340603510E-05,
+.28300608926316E-07,.50812598486162E-10,.45454545454545E+00,
+.39786982171073E+00,.25000002282553E-01,.59148858437864E-03,
+.66757562996758E-05,.38720468964770E-07,.11144761163396E-09/
 DATA D(121),D(122),D(123),D(124),D(125),D(126),D(127),
+D(128),D(129),D(130),D(131),D(132),D(133),D(134),D(135),
+D(136),D(137),D(138),D(139),D(140),D(141),D(142),D(143),
+D(144),D(145),D(146),D(147),D(148),D(149),D(150)/
+.12608153728000E-12,.45454545454545E+00,.39822650679417E+00,
+.25183525013803E-01,.60880314013113E-03,.72358433014328E-05,
+.46725317100106E-07,.16719594503501E-09,.31157527742621E-12,
+.23590370482110E-15,.45454545454545E+00,.39824868901869E+00,
+.25265819897767E-01,.61910411544859E-03,.76074711148850E-05,
+.52536344101559E-07,.21317642667569E-09,.50421345500406E-12,
+.64317688539028E-15,.34187163161474E-18,.45454545454545E+00,
+.39826570956750E+00,.25287844091655E-01,.62494425734623E-03,
+.78540296938450E-05,.56743151456408E-07,.24973869169769E-09,
+.68066567167935E-12,.11216260688042E-14,.10236802978533E-17/
 DATA D(151),D(152),D(153),D(154),D(155),D(156),D(157),
+D(158),D(159),D(160),D(161),D(162),D(163),D(164),D(165),
+D(166),D(167),D(168),D(169),D(170),D(171),D(172),D(173),
+D(174),D(175),D(176)/
+.39720657964596E-21,.45454545454545E+00,.39804225208567E+00,
+.25341708058167E-01,.63151501325496E-03,.80807554649258E-05,
+.60354664402269E-07,.28111642092611E-09,.84106334638657E-12,
+.16167280397671E-14,.19299941789907E-17,.13021732956413E-20,
+.37944453664700E-24,.45454545454545E+00,.39838448104318E+00,
+.25416762495946E-01,.63697712076076E-03,.82551983508228E-05,
+.63149729888770E-07,.30629858708888E-09,.97808681916440E-12,
+.20836563130596E-14,.29320679574331E-17,.26166497017204E-20,
+.13415328284784E-23,.30092796196223E-27/
 DATA D(177),D(178),D(179),D(180),D(181),D(182),D(183),D(184),
+D(185),D(186),D(187),D(188),D(189),D(190),D(191),D(192),D(193),
+D(194),D(195),D(196),D(197),D(198),D(199),D(200),D(201),D(202),
+D(203),D(204),D(205),D(206)/
+-.58064516129032E+00,-.21559395174672E+00,-.30544696579492E-01,
+-.58064516129032E+00,-.19115223031554E+00,-.29473834786102E-01,
+-.58064516129032E+00,-.18233307297138E+00,-.10066347258135E-02,
+-.58064516129032E+00,-.12030039601232E-02,-.15208008334815E-04,
+-.28236544473632E-01,-.12030039601232E-02,-.15208008334815E-04,
+-.58064516129032E+00,-.17833069941496E+00,-.28475246894827E-01,
+-.58064516129032E+00,-.17833069941496E+00,-.21343804115613E-06,
+-.14606302030482E-02,-.29964111364600E-04,-.21343804115613E-06,
+-.58064516129032E+00,-.17610367098315E+00,-.28260676645090E-01,
+-.58064516129032E+00,-.17610367098315E+00,-.28260676645090E-01,
+-.15322458810336E-02,-.36586320114212E-04,-.39867529427935E-06,
+-.16226665135199E-08,-.58064516129032E+00,-.17483390114911E+00,
+-.16226665135199E-08,-.58064516129032E+00,-.17483390114911E+00,
+-.27741186226246E-01,-.14815956989918E-02,-.36301045393729E-04/
+-.27741186226246E-01,-.14815956989918E-02,-.36301045393729E-04/
 DATA D(207),D(208),D(209),D(210),D(211),D(212),D(213),D(214),
+D(215),D(216),D(217),D(218),D(219),D(220),D(221),D(222),D(223),
+D(224),D(225),D(226),D(227),D(228),D(229),D(230),D(231),D(232),
+D(233),D(234),D(235),D(236)/
+-.44685007550778E-06,-.26875845072 81E-08,-.62831231083352E-11,
+-.58064516129032E+00,-.17396071288671E+00,-.28684134226593E-01,
+-.17432495919146E-02,-.50845860092993E-04,-.79137773126678E-06,
```

```
     +-.673747535477 29E-08,-.29589252318632E-10,-.52416294063507E-13,
     +-.580645161290 32E+00,-.17339878373842E+00,-.28044727272007E-01,
     +-.580645161290 32E+00,-.14813974810960 7E+01,
     +-.169789454510 1E+00,.15806451612903E+00,.14814748109607E+01,
     +.193160314147 98E+00,.62334163398843E-02,.15806451612903E+01,
     +.147265553616 5E+01,.20074218117966E+00,.86336976630508E-02,
     +.115808458719 7E-03,.15806451612903E+00,.15806532800601E+01,
     +.544583880157 84E-05,.56371977519294E-07,.37545083692440E-09,
     +.209163877038 69E+00,.11579454550239E-01,.32857305529008E-03,
     +.219776160728 10E-17,.15806451612903E+01,.14633254223081E+01,
     +.268550391116 66E-05,.10854850713601E-07,.15806451612903E+01,
     +.146515648179 43E+01,.20774599475456E+00,.10971861052267E-01,
     +.275243688300 2E-03,.35403656934423E-05,.22575321236761E-07/
     DATA D(297),D(298),D(299),D(300),D(301),D(302),D(303),D(304),
     +D(305),D(306),D(307),D(308),D(309),D(310),D(311),D(312),D(313),
     +D(314),D(315),D(316),D(317),D(318),D(319),D(320),D(321),D(322),
     +D(323),D(324),D(325),D(326)/
     +.565744878825 85E-10,.15806451612903E+01,.14642832935319E+01,
     +.209562131017 30E+00,.11509566107375E-01,.31097988163828E-03,
     +.456309861333 02E-05,.37079641130883E-07,.15682590950194E-09,
     +.269334300355 88E-12,.15806451612903E+01,.14637213643836E+01,
     +.656361463173 4E-08,-.42091025159364E-10,-.17479977056317E-12,
     +.453684068209 98E-15,-.66928703208412E-18,-.42844311155752E-21,
     +.580645161290 32E+00,-.17229366960984E+00,-.28800272381574E-01,
     +.185961744304 20E-02,-.59976978730906E-04,-.11090566945359E-05,
     +.127459356922 38E-07,-.95158694657869E-10,-.46969406476975E-12,
     +.152180641365 28E-14,-.31130952108508E-17,-.36466961532119E-20,
     +.186449343681 93E-23,.15806451612903E+01,.15059165323919E+01/
     DATA D(267),D(268),D(269),D(270),D(271),D(272),D(273),D(274),
     +D(275),D(276),D(277),D(278),D(279),D(280),D(281),D(282),D(283),
     +D(284),D(285),D(286),D(287),D(288),D(289),D(290),D(291),D(292),
     +D(293),D(294),D(295),D(296)/
     +.169789454510 1E+00,.15806451612903E+01,.14814748109607E+01,
     +.193160314147 98E+00,.62334163398843E-02,.15806451612903E+01,
     +.147265553616 5E+01,.20074218117966E+00,.86336976630508E-02,
     +.115808458719 7E-03,.15806451612903E+00,.15806532800601E+01,
     +.544583880157 84E-05,.56371977519294E-07,.37545083692440E-09,
     +.209163877038 69E+00,.11579454550239E-01,.32857305529008E-03,
     +.219776160728 10E-17,.15806451612903E+01,.14633254223081E+01,
     +.268550391116 66E-05,.10854850713601E-07,.15806451612903E+01,
     +.146515648179 43E+01,.20774599475456E+00,.10971861052267E-01,
     +.275243688300 2E-03,.35403656934423E-05,.22575321236761E-07/
     DATA D(237),D(238),D(239),D(240),D(241),D(242),D(243),D(244),
     +D(245),D(246),D(247),D(248),D(249),D(250),D(251),D(252),D(253),
     +D(254),D(255),D(256),D(257),D(258),D(259),D(260),D(261),D(262),
     +D(263),D(264),D(265),D(266)/
     +.170601377967 70E-12,-.35056663086597E-15,-.30628886618191E-18,
     +.580645161290 32E+00,-.17300284166294E+00,-.27328009024214E-01,
     +.150560215450 51E-02,-.41367453292114E-04,-.65884289780074E-06,
     +.455309861333 02E-05,.37079641130883E-07,.15682590950194E-09,
     +.269334300355 88E-12,.15806451612903E+01,.14637213643836E+01,
     +.209562131017 30E+00,.11509566107375E-01,.31097988163828E-03,
     +.565744878825 85E-10,.15806451612903E+01,.14642832935319E+01,
     +D(323),D(324),D(325),D(326)/
     +D(314),D(315),D(316),D(317),D(318),D(319),D(320),D(321),D(322),
     +D(305),D(306),D(307),D(308),D(309),D(310),D(311),D(312),D(313),
     DATA D(297),D(298),D(299),D(300),D(301),D(302),D(303),D(304),
     +.275243688300 2E-03,.35403656934423E-05,.22575321236761E-07/
     +.146515648179 43E+01,.20774599475456E+00,.10971861052267E-01,
     +.268550391116 66E-05,.10854850713601E-07,.15806451612903E+01,
     +.206995715339 35E+00,.10680275405545E-01,.24933405067263E-03,
     +.139192014489 22E-05,.15806451612903E+01,.14664262516283E+01,
     +.204983257157 28E+00,.99938112886329 1E-02,.19922348036296E-03,
```

```
     +.21134531244915E+00,.12108121568554E-01,.35894153745450E-03,
     +.62664422624671E-05,.69193362616297E-07,.50195179261835E-09,
     +.24253292036317E-11,.77310365461803E-14,.15613921810526E-16,
     +.18102819599155E-19,.91776107476727E-23/
C
      DATA D(353),D(354),D(355),D(356),D(357),D(358),D(359),D(360),
     +D(361),D(362),D(363),D(364),D(365),D(366),D(367),D(368),D(369),
     +D(370),D(371),D(372),D(373),D(374),D(375),D(376),D(377),D(378),
     +D(379),D(380),D(381),D(382),D(383),D(384),D(385),D(386),D(387),
     +D(388),D(389),D(390),D(391),D(392),D(393),D(394),D(395),D396),
     +D(397),D(398),D(399),D(400),D(401,D402/
     +1.,1.,.5,
     +1.,1.,0.5,63304085.E-09,1.,1.,0.5,79027358.E-09,37089254.E-10,
     +1.,1.,0.5,85605575.E-09,56773923.E-10,12758716.E-11,1.,1.,0.5,
     +89018496.E-09,67947003.E-10,23143226.E-11,2898388.E-12,1.,1.,0.5,
     +91025408.E-09,74822425.E-10,30567580.E-11,6072001.E-12,
     +4679323.E-14,1.,1.,0.5,92308224.E-09,79335426.E-10,35849723.E-11,
     +8800161.E-12,11108474.E-14,5648779.E-16,1.,1.,0.5,93178948.E-09,
     +82451157.E-10,39680345.E-11,11000301.E-12,17552367.E-14/
      DATA D(403),D(404),D(405),D(406),D(407),D(408),D(409),D(410),
     +D(411),D(412),D(413),D(414),D(415),D(416),D(417),D(418),D(419),
     +D(420),D(421),D(422),D(423),D(424),D(425),D(426),D(427),D(428),
     +D(429),D(430),D(431),D(432),D(433),D(434),D(435),D(436)D(37),
     +D(438),D(439),D(440)/
     +14976734.E-16,5293311.E-18,1.,1.,0.5,93797465.E-09,84690176.E-10,
     +42524183.E-11,12746945.E-12,23355062.E-14,25642831.E-16,
     +15495967.E-18,3962808.E-22,1.,1.,0.5,94252811.E-09,86352191.E-10,
     +44683802.E-11,14135170.E-12,28359079.E-14,36242802.E-16,
     +28591262.E-18,12691526.E-20,24249737.E-23,1.,1.,0.5,94597848.E-09,
     +87619296.E-10,46357872.E-11,15246910.E-12,32599754.E-14,
     +46107927.E-16,42819928.E-18,25110371.E-20,84319465.E-23,
     +12357105.E-25/
C
      ORDER=INFO(12)
      M=INFO(9)
      IF(INFO(13).LT.2) GOTO 50
      M1=176*(ORDER-1)+M*(M+1)/2-3
      M2=88+M1
      MM=M+1
      DO 10 J=1,MM
      M1PJ=M1+J
      M2PJ=M2+J
      P(J)=D(M1PJ)
      S(J)=D(M2PJ)
   10 CONTINUE
      IF(M.GT.2) GOTO 20
      P(4)=0.0
      S(4)=0.0
   20 DD=1.375-.6*FLOAT(ORDER-1)
      E=P(2)+2.0*(-P(3)+P(4)+S(4))
      C(M)=(E/DD-((DD-0.5)/DD)**2)/(2.+E)
      LA(M)=1.0/DD-C(M)
      LA(M-1)=S(3)/LA(M)
      C(M-1)=P(3)/LA(M)
      B(1)=(P(2)-C(M))/LA(M)
      B(2)=P(1)
      IF(M.GT.2) GOTO 30
```

```
      RETURN
C
   30 MM=M-2
      DO 40 J=1,MM
      MP2MJ=M+2-J
      MP1MJ=M+1-J
      C(J)=P(MP2MJ)/S(MP1MJ)
      LA(J)=S(MP2MJ)/S(MP1MJ)
   40 CONTINUE
      RETURN
C
   50 M1=352+M*(M+1)/2-3
      MM=M+1
      DO 60 J=1,MM
      M1PJ=M1+J
   60 S(J)=D(M1PJ)
      B(1)=0.0
      B(2)=0.0
      C(M)=0.0
      LA(M)=S(1)
      MM=M-1
      DO 70 J=1,MM
      MP2MJ=M+2-J
      MP1MJ=M+1-J
      LA(J)=S(MP2MJ)/S(MP1MJ)
      C(J)=0
   70 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE POWERM(N,Y,Y1,YXE,DY,DY1,F,SIGMA,APR,IFLAG,INFO)
C*****************************************************************
C* IF ON ENTRY SIGMA(1)=0 POWERM COMPUTES AN ESTIMATION OF THE SPECTR- *
C* AL RADIUS OF THE JACOBIAN BY MEANS OF AN ADJUSTED POWER METHOD.THE  *
C* ITERATION IS STOPPED IF TWO SUCCESSIVE ITERATES DIFFER RELATIVELY   *
C* LESS THAN 0.001.THE MINIMAL NUMBER OF ITERATIONS IS 5.IF THE COMPU- *
C* TATION DID NOT SUCCEED WITHIN 50 ITERATIONS AN ERRORMESSAGE IS GIV- *
C* EN.AS A SAFETY MARGIN,THE LAST ITERATE IS ENLARGED WITH 10 PERCENT. *
C* THE RESULT IS STORED IN SIGMA(1).                                   *
C*                                                                     *
C* IF ON ENTRY SIGMA(1) NOT EQUALS 0 POWERM PERFORMS THREE ITERATIONS  *
C* WITH THE ADJUSTED POWER METHOD.IF THE THIRD ITERATE IS MORE THAN 10 *
C* PERCENT SMALLER THAN SIGMA(2),THE ITERATION IS CONTINUED AS IF      *
C* SIGMA(1)=0.IN THIS CASE THE THIRD ITERATE IS STORED IN SIGMA(2).    *
C*                                                                     *
C* THE POWER METHOD REQUIRES THREE WORK ARRAYS.WE USE YXE,DY AND DY1,  *
C* WHERE DY AND DY1 ARE OVERWRITTEN.ON ENTRY OF THIS ROUTINE DY AND    *
C* DY1 CONTAIN F(Y) AT Y=Y(X) AND Y=Y(X-H),RESPECTIVELY.THUS ON EXIT   *
C* OF THIS ROUTINE TWO EXTRA EVALUATIONS OF F ARE NECESSARY FOR RESTO- *
C* RING THE DERIVATIVES.                                               *
C*                                                                     *
C* THIS CODE USES THE CDC SYSTEM SUBPROGRAMS RANSET AND RANF,GENE-     *
C* RATING RANDOM VALUES FROM THE INTERVAL [0,1]. THE ARGUMENT OF RANF  *
C* IS DUMMY AND IGNORED.RANSET INITIALIZES THE GENERATIVE VALUE OF RANF*
C*****************************************************************
      DIMENSION Y(N),Y1(N),YXE(N),DY(N),DY1(N),SIGMA(2),INFO(15)
      REAL NORM,NORM0
      IF(INFO(2).EQ.3) INFO(2)=1
      INFO(15)=0
      TOLLIP=1.E+4*APR
      SIGM=0.0
      S0=0.0
      CALL RANSET(0)
      DO 10 I=1,N
      RA=2.*RANF(IDUM)-1.
      YXE(I)=DY(I)
      IF(Y(I).EQ.0.0) DY(I)=RA*TOLLIP
      IF(Y(I).NE.0.0) DY(I)=Y(I)*(1.0+TOLLIP*RA)
      DY1(I)=DY(I)
      S0=S0+DY(I)*DY(I)
   10 CONTINUE
      NORM0=TOLLIP*SQRT(S0)
      IF(NORM0.LT.TOLLIP)NORM0=TOLLIP
      CALL F(N,DY1)
      INFO(7)=INFO(7)+1
      INFO(8)=INFO(8)+1
C
      DO 70 K=1,51
      IF(K.LT.51) GOTO 20
      IFLAG=3
      RETURN
C
   20 S0=0
      DO 30 I=1,N
      S1=YXE(I)-DY1(I)
      S0=S0+S1*S1
   30 CONTINUE
```

```
      NORM=SQRT(S0)
      SIGM1=SIGM
      SIGM=NORM/NORM0
C
      IF(K.EQ.3.AND.SIGMA(1).EQ.0.0) SIGMA(2)=SIGM
      IF(K.LE.2.OR.SIGMA(1).EQ.0.0) GOTO 40
      IF(SIGM.GE.0.9*SIGMA(2)) GOTO 80
      SIGMA(2)=SIGM
      SIGMA(1)=0.
C
   40 IF(ABS(SIGM1-SIGM)/SIGM.GT.0.001.OR.K.LE.4) GOTO 50
      SIGMA(1)=1.1*SIGM
      GOTO 80
   50 DO 60 I=1,N
   60 YXE(I)=DY(I)+(YXE(I)-DY1(I))/SIGM
      CALL F(N,YXE)
      INFO(7)=INFO(7)+1
      INFO(8)=INFO(8)+1
   70 CONTINUE
C
   80 DO 90 I=1,N
   90 DY(I)=Y(I)
      CALL F(N,DY)
      INFO(7)=INFO(7)+1
      INFO(8)=INFO(8)+1
      IF(INFO(13).NE.0) GOTO 100
      RETURN
  100 DO 110 I=1,N
  110 DY1(I)=Y1(I)
      CALL F(N,DY1)
      INFO(7)=INFO(7)+1
      INFO(8)=INFO(8)+1
      RETURN
      END


      SUBROUTINE MAXDEG(TOL,APR,IFLAG,INFO)
C*****************************************************************
C* IN MAXDEG THE MAXIMAL DEGREES WITH RESPECT TO THE INTERNAL STABILI- *
C* TY CONDITION ARE COMPUTED. IF ONE OF THESE MAXIMAL DEGREES FALLS    *
C* OUTSIDE THE RANGE,AN ERRORMESSAGE IS GIVEN.                         *
C*****************************************************************
      DIMENSION Q(11),INFO(15)
      DATA Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8),Q(9),Q(10),Q(11)/
     +3.E1,1.E2,7.E2,4.E3,3.E4,2.E5,9.E5,5.E6,3.E7,2.E8,1.E9/
      E=TOL/APR
      DO 10 I=2,12
      J=13-I
      IF(Q(J).LE.E) GOTO 20
   10 CONTINUE
      IFLAG=2
      RETURN
C
   20 INFO(10)=14-I
      DO 30 I=2,12
      J=13-I
      IF(100.0*Q(J).LE.E) GOTO 40
   30 CONTINUE
      IFLAG=2
      RETURN
C
   40 INFO(11)=14-I
      RETURN
      END
```

```
      SUBROUTINE MINDEG(H,SIGMA,INFO)
C*****************************************************************
C* MINDEG DETERMINES THE MINIMAL DEGREE M WHICH STILL GIVES RISE TO A  *
C* STABLE INTEGRATION STEP.                                            *
C*****************************************************************
      DIMENSION INFO(15)
      LOGICAL START
      L=INFO(12)+9
      MMAX=INFO(L)
      START=INFO(13).LT.2
      IF(.NOT.START) BETAC=5.15+2.86*FLOAT(1-INFO(12))
      DO 10 M=2,MMAX
      RM=M
      IF(START) BETAC=0.03*RM+0.44
      IF(H.LE.BETAC*RM*RM/SIGMA) GOTO 20
   10 CONTINUE
      M=MMAX
   20 INFO(9)=M
      RETURN
      END
```

```
      SUBROUTINE STEP(N,Y,Y1,Y2,YXE,DY,DY1,H,F,C,LA,B,INFO)
C*****************************************************************
C* STEP CONTAINS THE ACTUAL INTEGRATOR.FOR CONVENIENCE,THE ONE-STEP    *
C* SCHEME IS ALSO FORMULATED AS A THREE-STEP SCHEME BY INTRODUCING ZE- *
C* RO-PARAMETERS.ON EXIT OF THIS ROUTINE THE NEW CALCULATED SOLUTION   *
C* VECTOR IS CONTAINED  IN YXE.                                        *
C*****************************************************************
      REAL LA
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N),DY(N),DY1(N),
     +LA(12),C(12),B(2),INFO(15)
      M=INFO(9)
      D=1.375-.6*FLOAT(INFO(12)-1)
      IF(INFO(13).LT.2) D=1.0
      DO 10 I=1,N
   10 YXE(I)=DY(I)
      IF(M.EQ.2) GOTO  40
      MM=M-2
C
      DO 30 J=1,MM
      HC=H*C(J)
      HLA=H*LA(J)
      DO 20 I=1,N
      YXE(I)=Y(I)+HC*DY1(I)+HLA*YXE(I)
   20 CONTINUE
      CALL F(N,YXE)
      INFO(7)=INFO(7)+1
   30 CONTINUE
C
   40 BM1=B(1)
      BM11=1.0-BM1
      HC=H*C(M-1)
      HLA=H*LA(M-1)
      DO 50 I=1,N
   50 YXE(I)=BM11*Y(I)+BM1*Y1(I)+HC*DY1(I)+HLA*YXE(I)
      D1=1.-D
      BM1=B(2)*D
      BM11=(1.-B(2))*D
      HC=H*C(M)*D
      HLA=H*LA(M)*D
      CALL F(N,YXE)
      INFO(7)=INFO(7)+1
      DO 60 I=1,N
   60 YXE(I)=BM11*Y(I)+BM1*Y1(I)+D1*Y2(I)+HC*DY1(I)+HLA*YXE(I)
      RETURN
      END
```

```
      SUBROUTINE ESTIMA(N,Y,Y1,Y2,YXE,TOL,EPS,ERROR,INFO)
C*****************************************************************
C* ESTIMA CALCULATES THE LOCAL ERROR BOUND EPS=(1+NORM(Y))*TOL FOR THE *
C* MIXED ERROR TEST AND ESTIMATES THE LOCAL ERROR ERROR.          *
C*****************************************************************
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N),INFO(15),CONST(2)
      INTEGER ORDER
      CONST(1)=2.85
      CONST(2)=0.49
      ORDER=INFO(12)
      EPS=0.0
      ERROR=0.0
      IF(ORDER.EQ.2) GOTO 20
      DO 10 I=1,N
      YI=Y(I)
      EPS=EPS+YI*YI
      E=Y1(I)-YI-YI+YXE(I)
      ERROR=ERROR+E*E
   10 CONTINUE
C
      GOTO 40
   20 DO 30 I=1,N
      EPS=EPS+Y(I)*Y(I)
      E=-Y2(I)+3.0*(Y1(I)-Y(I))+YXE(I)
      ERROR=ERROR+E*E
   30 CONTINUE
C
   40 EPS=TOL+TOL*SQRT(EPS/FLOAT(N))
      ERROR=CONST(ORDER)*SQRT(ERROR/FLOAT(N))
      RETURN
      END



      SUBROUTINE NEWH(EPSERR,HOLD,H,ALFA,HMAX,INFO)
C*****************************************************************
C* NEWH DELIVERS A NEW STEPLENGTH AND THE FACTOR ALFA BY WHICH THE  *
C* STEPLENGTH IS CHANGED. EPSERR DENOTES EPS/ERROR.               *
C*****************************************************************
      DIMENSION HMAX(2),INFO(15)
      INTEGER ORDER
      ALFA=1.0
      IF(INFO(14).GE.3.OR.EPSERR.LE.1.0) GOTO 10
      RETURN
C
   10 ORDER=INFO(12)
      ALFA=EPSERR**(1./FLOAT(ORDER+1))/(2.0-FLOAT(ORDER-1)*0.4)
      IF(ALFA.GT.0.9.AND.ALFA.LT.1.1) ALFA=1.0
      IF(ALFA.NE.1.0) GOTO 20
C
      RETURN
   20 IF(ALFA.GT.3.0) ALFA=3.0
      IF(ALFA.LT.0.1) ALFA=0.1
      H=HOLD*ALFA
      IF(H.GT.HMAX(ORDER)) H=HMAX(ORDER)
      ALFA=H/HOLD
      RETURN
      END
```

```
      SUBROUTINE INTER1(N,Y,Y1,Y2,DY1,F,ALFA,INFO)
C*****************************************************************
C* INTER1 COMPUTES VALUES FOR Y(X-ALFA*H) AND Y(X-2*ALFA*H) FROM Y(X), *
C* Y(X-H) AND Y(X-2H) BY QUADRATIC INTERPOLATION,AND COMPUTES THE DER- *
C* IVATIVE AT X-H BY CALLING F.                                  *
C*****************************************************************
      DIMENSION Y(N),Y1(N),Y2(N),DY1(N),INFO(15)
      REAL NU
      NU=2.-ALFA
      C12=(NU-1.)*(NU-2.)/2.
      C11=NU*(2.-NU)
      C10=NU*(NU-1.)/2.
      NU=2.-2.*ALFA
      C22=(NU-1.)*(NU-2.)/2.
      C21=NU*(2.-NU)
      C20=NU*(NU-1.)/2.
C
      DO 10 I=1,N
      CY0=Y(I)
      CY1=Y1(I)
      CY2=Y2(I)
      Y1(I)=C12*CY2+C11*CY1+C10*CY0
      DY1(I)=Y1(I)
      Y2(I)=C22*CY2+C21*CY1+C20*CY0
   10 CONTINUE
      CALL F(N,DY1)
      INFO(7)=INFO(7)+1
      INFO(14)=0
      RETURN
      END
```

```
      SUBROUTINE INTER2(N,Y,Y1,Y2,YXE,A)
C*****************************************************************
C* INTER2 COMPUTES THE SOLUTION AT THE OUTPUT POINT XE=X-A*H BY QUAD-  *
C* RATIC INTERPOLATION BETWEEN Y(X),Y(X-H) AND Y(X-2H).THE RESULT IS   *
C* STORED IN YXE.                                                *
C*****************************************************************
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N)
      REAL NU
      NU=2.-A
      C12=(NU-1.)*(NU-2.)/2.
      C11=NU*(2.-NU)
      C10=NU*(NU-1.)/2.
      DO 10 I=1,N
   10 YXE(I)=C12*Y2(I)+C11*Y1(I)+C10*Y(I)
      RETURN
      END
```

```
      SUBROUTINE SHIFT(N,Y,Y1,Y2,YXE,DY,DY1,X,H,F,INFO)
C*****************************************************************
C* SHIFT SHIFTS THE X AND Y VARIABLES AND COMPUTES F(Y(X+H))          *
C*****************************************************************
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N),DY(N),DY1(N),INFO(15)
      DO 10 I=1,N
      Y2(I)=Y1(I)
      Y1(I)=Y(I)
      Y(I)=YXE(I)
      DY1(I)=DY(I)
      DY(I)=Y(I)
   10 CONTINUE
      CALL F(N,DY)
      INFO(7)=INFO(7)+1
      X=X+H
      RETURN
      END
```

SAMENVATTING

In dit proefschrift beschouwen we beginwaardeproblemen voor stelsels gewone differentiaalvergelijkingen

$$\frac{d\vec{y}}{dx} = \vec{f}(x,\vec{y}).$$

Deze problemen komen veelvuldig voor in de wiskunde en natuurwetenschappen, doch men kan ze zelden exact oplossen. Het bepalen van een benaderende, en voor de praktijk voldoend nauwkeurige oplossing, is echter bijna altijd mogelijk door middel van numerieke integratietechnieken. Numerieke integratie is een belangrijk onderdeel van de numerieke wiskunde, waarvoor de basisideeën al ontwikkeld zijn aan het begin van deze eeuw. Net zoals bij andere onderdelen van de numerieke wiskunde, is de ontwikkeling van de numerieke integratie echter pas goed op gang gekomen sinds de komst van de computer, waarmee de technieken gemakkelijk en efficiënt te realiseren zijn. Dit proefschrift is samengesteld uit vier wetenschappelijke publikaties over dit onderwerp. Deze publikaties worden genoemd in het voorwoord.

Zeer bekende en veelvuldig toegepaste technieken zijn gebaseerd op de klassieke Runge-Kuttaformules en de klassieke meerstapsformules. Deze technieken kan men met vrucht gebruiken indien de Lipschitzconstante

$$L = \sup \| \partial \vec{f} / \partial \vec{y} \| ,$$

genomen over de punten $(x,\vec{y})$ langs de oplossingskromme, "voldoende klein" is. Daarentegen worden deze formules inefficiënt als L "groot" is. In dat geval moet men met "zeer kleine" integratiestappen integreren om het proces numeriek stabiel te houden, wat in een enorme rekentijd op de computer kan resulteren. Voor beginwaardeproblemen met een grote Lipschitzconstante zijn daarom speciale integratieformules nodig.

In het proefschrift beschouwen we in het bijzonder twee klassen van zulke problemen, namelijk beginwaardeproblemen voor stijve differentiaalvergelijkingen en voor semi-gediscretiseerde parabolische differentiaalvergelijkingen. Een bekend voorbeeld van een stijve vergelijking is de vergelijking welke een aantal gekoppelde chemische reacties beschrijft

waarbij sommige reacties vele malen sneller verlopen dan andere. Deze vergelijking zal de eigenschap hebben dat de oplossing zowel snel variërende als langzaam variërende componenten bevat, welke duidelijk te onderscheiden zijn. Zo'n vergelijking noemt men stijf. Een bekend voorbeeld van een parabolische vergelijking is de vergelijking welke de warmtegeleiding van een of ander ruimtelijk medium, zoals een lange metalen staaf, beschrijft. Het discretiseren van de plaatsvariabelen in deze partiële differentiaalvergelijking, d.w.z. het semi-discretiseren, geeft een vergelijking van het type welke wij beschouwen. Zo'n semi-gediscretiseerde vergelijking bevat ook snel variërende componenten, net zoals een stijve vergelijking. Deze snel variërende componenten zorgen voor de grote Lipschitzconstante.

Voor de stijve vergelijkingen worden twee klassen van integratieformules behandeld, namelijk gegeneraliseerde lineaire meerstapsformules en gegeneraliseerde Runge-Kuttaformules. Een gegeneraliseerd schema verschilt van een klassiek schema hierin, dat de coëfficiënten van de formules matrices zijn in plaats van scalairen. Deze matrices hangen af van de Jacobiaan van het te integreren stelsel en worden gegenereerd door rationale functies. Door deze rationale functies te introduceren is het mogelijk om de numerieke stabiliteitsbeperkingen geheel op te heffen. In de eerste publikatie komen de meerstapsformules aan de orde, terwijl de tweede publikatie de Runge-Kuttaformules behandelt. In deze tweede publikatie wordt met name de stabiliteit van de formules voor niet-lineaire stelsels onderzocht.

In de derde publikatie wordt een klasse van integratieformules gegeven voor de semi-gediscretiseerde parabolische vergelijkingen. Dit zijn expliciete drie-staps Runge-Kuttaformules met vergrote reële stabiliteitsgrenzen, Het voordeel van deze formules is dat ze makkelijk te gebruiken zijn voor een grote klasse van vergelijkingen. Een nadeel is dat de numerieke stabiliteitsbeperking voor de integratiestap niet geheel ongedaan gemaakt kan worden, vanwege het expliciet zijn. Aan deze beperking wordt echter in belangrijke mate tegemoet gekomen door de vergrote stabiliteitsgrenzen. Een volledig gedocumenteerd en overdraagbaar FORTRAN-programma, gebaseerd op deze gestabiliseerde formules, en voorzien van fout-en staplengtecontrole wordt besproken in de vierde publikatie. Hierin worden tevens numerieke voorbeelden gepresenteerd.

SOR

ONTVANGEN 18 OKT. 1977

AMS(CHOS) 6SLOS
6SM20
00.08

STELLINGEN

BIJ HET PROEFSCHRIFT

ON THE CONSTRUCTION AND ANALYSIS OF STABLE NUMERICAL METHODS FOR STIFF AND PARABOLIC DIFFERENTIAL EQUATIONS

VAN

J.G. VERWER

23 November 1977

I

Laat P en S polynomen van de graad $m \geq 2$ zijn, gegeven door

$$P(x) = -1 + \sum_{i=1}^{m} p_i x^i, \qquad p_i \in \mathbb{R},$$

$$S(x) = 2 - p_1 x + (1+p_1-p_2)x^2 + \sum_{i=3}^{m} s_i x^i, \qquad s_i \in \mathbb{R}.$$

Het interval $[-\beta,0]$, waarop voldaan is aan de ongelijkheden $|S(x)| \leq 1 - P(x)$, $|P(x)| \leq 1$, wordt dan gemaximaliseerd door de keuze

$$P(x) = -\cos\left(m \arccos\left(1 + \frac{x}{m^2}\right)\right), \quad S(x) = 1 - P(x),$$

waarvoor $\beta = 2m^2$.

VERWER, J.G., *Multipoint multistep Runge-Kutta methods I: On a class of two-step methods for parabolic equations*, Report NW 30/76, Mathematisch Centrum, Amsterdam, 1976.

II

Een gegeneraliseerd Runge-Kuttaschema welke behoort tot de door ROSENBROCK voorgestelde klasse, en welke een sterk A-stabiele stabiliteitsfunctie bezit, is altijd S-stabiel.

ROSENBROCK, H.H., *Some general implicit processes for the numerical solution of differential equations*, Comput. J. 5, 329-330, 1963.

III

De in het proefschrift behandelde adaptieve meerstapsformule kan beschouwd worden als een generalisatie van de bekende expliciete Adams-Bashforth-formule. De equivalente generalisatie van de impliciete Adams-Moultonformule luidt

$$\vec{y}_{n+1} = R(h_n J_n)\vec{y}_n + \sum_{\ell=0}^{k} h_n B_\ell (h_n J_n)[\vec{y}'_{n+1-\ell} - J_n \vec{y}_{n+1-\ell}].$$

Voor willekeurige matrices J_n valt te bewijzen dat dit schema consistent is

van de orde k+1, indien $R(z) = e^z + O(z^{k+2})$, $z \to 0$, en

$$\sum_{\ell=0}^{k} q_{\ell-1}^{j-1} B_\ell(z) = D_j(z) \quad , \qquad j = 1,\ldots,k+1,$$

$$D_1(z) = \frac{R(z)-1}{z}, \quad D_{j+1}(z) = \frac{jD_j(z)-1}{z}, \qquad j = 1,\ldots,k,$$

$$q_\ell = (x_{n-\ell} - x_n)/h_n \quad , \qquad \ell = -1,0,\ldots,k.$$

VAN DER HOUWEN, P.J. & VERWER J.G., *Generalized linear multistep methods I: Development of algorithms with zero-parasitic roots*, Report NW 10/74, Mathematisch Centrum, Amsterdam, 1974.

IV

De gegeneraliseerde Adamsformules voor stijve vergelijkingen $\vec{y}' = \vec{f}(x,\vec{y})$ kan men ook afleiden door:

a) Lawson-equilibratie toepassen, d.w.z. het rechterlid lineair opsplitsen in een stijf gedeelte $J\vec{y}$, J benadering van de Jacobiaan, en een niet-stijf gedeelte $\vec{f}(x,\vec{y}) - J\vec{y}$.

b) De oplossing $\vec{y}$ in x+h representeren via de integraalvergelijking

$$\vec{y}(x+h) = e^{hJ}\vec{y}(x) + h\int_0^1 e^{(1-\tau)hJ}[y'(x+\tau) - Jy(x+\tau)]d\tau.$$

c) De kwadratuurregels die corresponderen met de Adamsformules toepassen.

d) Een geschikte approximatie kiezen voor de exponentiële functie via een polynoom of een rationale functie.

LAWSON, J.D., *Some numerical methods for stiff ordinary and partial differential equations*, Proc. Second Manitoba Conference on Numerical Math., 27-34, 1972.

LEGRAS, J., *Résolution numérique des grands systèmes différentiels linéaires*, Numer. Math. 8, 14-28, 1966.

V

Beschouw het stelsel gewone differentiaalvergelijkingen $\vec{y}' = \vec{f}(x,\vec{y})$ en het eenstapsschema

$$\vec{y}_{n+\frac{1}{2}} = \vec{y}_n + \tfrac{1}{2}h_n \Lambda_E \vec{f}(x_n,\vec{y}_n) + \tfrac{1}{2}h_n \Lambda_0 \vec{f}(x_{n+\frac{1}{2}},\vec{y}_{n+\frac{1}{2}}),$$

$$\vec{y}_{n+1} = \vec{y}_{n+\frac{1}{2}} + \tfrac{1}{2}h_n \Lambda_0 \vec{f}(x_{n+\frac{1}{2}},\vec{y}_{n+\frac{1}{2}}) + \tfrac{1}{2}h_n \Lambda_E \vec{f}(x_{n+1},\vec{y}_{n+1}),$$

waarin Λ_E en Λ_0 vierkante matrices zijn. Dit schema is consistent van de orde 2 indien $\Lambda_E + \Lambda_0 = I$, I de eenheidsmatrix. Indien verder geldt dat Λ_E een diagonaalmatrix is waarvoor

$$(i,i)\text{-element van } \Lambda_E = \begin{cases} 1, & i \text{ even}, \\ 0, & i \text{ oneven}, \end{cases}$$

dan is dit schema identiek aan het "*odd-even hopscotch*" schema, mits $\vec{f}$ verkregen is uit semi-discretisering van een parabolische differentiaalvergelijking en voldoet aan de in GOURLAY genoemde "*E-property*". Het "*odd-even hopscotch*"-schema, dat ontwikkeld is als een directe methode voor partiële vergelijkingen, kan aldus toegepast worden via de methode der lijnen.

GOURLAY, A.R., *Hopscotch, a fast second order partial differential equation solver*, J. Inst. Math. Applics. 7, 216-227, 1971.

VERWER, J.G., *A comparison between the odd-even hopscotch method and a class of Runge-Kutta methods with extended real stability intervals*, Report NN 13/77, Mathematisch Centrum, Amsterdam, 1977.

VI

Zij gegeven een 5-punts gekoppeld stelsel gewone differentiaalvergelijkingen, verkregen uit semi-discretisering van een twee-dimensionale parabolische differentiaalvergelijking, in de componentsgewijze matrixnotatie

$$u'_{ij} = f_{ij}(t,u_{ij-1},u_{i-1j},u_{ij},u_{i+1j},u_{ij+1}).$$

Het componentsgewijs gedefinieerde, 2de orde eenstapsschema

$$u_{ij}^{n+\frac{1}{2}} = u_{ij}^{n} + \tfrac{1}{2}\tau_n f_{ij}\left(t_n + \alpha_1\tau_n, u_{ij-1}^{n}, u_{i-1j}^{n+\frac{1}{2}}, \tfrac{1}{2}\left(u_{ij}^{n} + u_{ij}^{n+\frac{1}{2}}\right), u_{i+1j}^{n+\frac{1}{2}}, u_{ij+1}^{n}\right),$$

$$u_{ij}^{n+1} = u_{ij}^{n+\frac{1}{2}} + \tfrac{1}{2}\tau_n f_{ij}\left(t_n + \alpha_2\tau_n, u_{ij-1}^{n+1}, u_{i-1j}^{n+\frac{1}{2}}, \tfrac{1}{2}\left(u_{ij}^{n+\frac{1}{2}} + u_{ij}^{n+1}\right), u_{i+1j}^{n+\frac{1}{2}}, u_{ij+1}^{n+1}\right),$$

waarin $\tau_n = t_{n+1} - t_n$, $\alpha_1 + \alpha_2 = 1$, is voor lineaire vergelijkingen te herkennen als een *ADI-methode* van het type van PEACEMAN en RACHFORD. Door middel van de methode der lijnen kan men aldus de *ADI-methode* definiëren voor, en toepassen op, willekeurige niet-lineaire problemen. Hierbij is het opsplitsbaar zijn van de differentiaaloperator niet noodzakelijk.

PEACEMAN, D.W. & H.H. RACHFORD JNR., *The numerical solution of parabolic and elliptic differential equations*, J. Soc. Ind. Appl. Math. 3, 28-41, 1955.

VII

Voor de in stelling VI beschouwde problemen, waarbij een willekeurige niet-lineaire koppeling toegelaten is, bestaat geen consistente methode welke voor lineaire problemen equivalent is aan de door YANENKO gegeven *LOD-methode*. Zo'n *LOD-methode* is wel consistent indien f_{ij} lineair op te splitsen is in de vorm

$$f_{ij}(t, u_{ij-1}, u_{i-1j}, u_{ij}, u_{i+1j}, u_{ij+1}) =$$

$$f_{ij}^{i}(t, u_{i-1j}, u_{ij}, u_{i+1j}) + f_{ij}^{j}(t, u_{ij-1}, u_{ij}, u_{ij+1}).$$

YANENKO, N.N., *The method of fractional steps*, Springer-Verlag, Berlin, 1971.

VAN DER HOUWEN, P.J. & J.G. VERWER, *A general formulation of linear splitting methods for ordinary and partial differential equations*, Report NW 47/77, Mathematisch Centrum, Amsterdam, 1977.

VIII

In de numerieke analyse van tijdsafhankelijke, partiële differentiaalvergelijkingen biedt de methode der lijnen een aantal duidelijke voordelen boven de directe methode. In het algemeen worden deze voordelen onvoldoende onderkend. Dit geldt zowel voor de constructie, de analyse, als voor de implementatie van de numerieke formules.

IX

De in ABRAMOWITZ & STEGUN opgenomen formule 6.5.33, namelijk

$$\gamma(a,z) \sim \sum_{n=0}^{\infty} \frac{(-1)^n z^{a+n}}{(a+n)n!}, \qquad a \to +\infty,$$

geeft geen korrekte asymptotische ontwikkeling voor de incomplete gamma-functie γ. De formule

$$\gamma(a,z) \sim \Gamma(a)e^{-z} \sum_{n=0}^{\infty} \frac{z^{a+n}}{\Gamma(a+n+1)}, \qquad a \to +\infty,$$

is wel korrekt.

ABRAMOWITZ, M. & I.A. STEGUN, *Handbook of mathematical functions*, National Bureau of Standards Applied Mathematics Series 55, U.S. Government Printing Office, Washington, 1964.

X

Het verdient aanbeveling dat wetenschappelijke instellingen waar in het Engels gepubliceerd wordt, een corrector in dienst nemen om het niveau van het Engels te verhogen. Deze corrector dient vertrouwd te zijn met het vakjargon van het betreffende wetenschapsgebied.

BOOMGAART, L., *De Engelse taal misbruikt. Over fouten in vertalingen van wetenschappelijke publikaties*, Intermediair, No 47, 1976.

De bewering van JURKAT en PEYERIMHOFF*), dat het uit 1897 daterende en heden ten dage in twijfel getrokken *"Vermoeden van Mertens"*,

$$\left| \sum_{n \leq x} \mu(n) \right| < \sqrt{x}, \quad x > 1, \quad \mu \text{ de Möbiusfunctie},$$

misschien ontzenuwd zou kunnen worden met behulp van een moderne snelle computer, vertoont een analogie met de door MAU TSE-TOENG aangehaalde parabel *"Hoe Joekong de bergen verplaatste"*.

JURKAT, W. & A. PEYERIMHOFF, *A constructive approach to Kronecker approximations and its application to the Mertens conjecture*, J. Reine Angew. Math., Band 286/287, 322-340, 1976.

MAU TSE-TOENG, *Gedachten en Gedichten*, In de serie "Manifesten" van L.J.C. Boucher, Den Haag, 1970.

---

*) De bewering van Jurkat en Peyerimhoff is gebaseerd op numerieke experimenten uitgevoerd met de WANG 700-tafelcomputer.