THEORETICAL AND COMPUTATIONAL ASPECTS OF THE NUMERICAL INTEGRATION OF HYPERBOLIC 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. J. BRUYN,
HOOGLERAAR IN DE FACULTEIT DER LETTEREN,
IN HET OPENBAAR TE VERDEDIGEN
IN DE AULA DER UNIVERSITEIT
(TIJDELIJK IN DE LUTHERSE KERK, INGANG SINGEL 411, HOEK SPUI)
OP WOENSDAG 3 SEPTEMBER 1980 DES NAMIDDAGS TE 13.30 UUR

DOOR

KEES DEKKER

GEBOREN TE MEDEMBLIK

1980
UNIVERSITEIT VAN AMSTERDAM

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

COPROMOTOR: PROF. DR. T.J. DEKKER

COREFERENT: PROF. DR. J.C. BUTCHER

Druk: Centrale Drukkerij Universiteit van Amsterdam

VOORWOORD

Het onderzoek waarvan de resultaten in dit proefschrift zijn neergelegd, is verricht in de Vakgroep Informatica en Numerieke Wiskunde van de Universiteit van Amsterdam, op de afdeling Numerieke Wiskunde van het Mathematisch Centrum en op de Onderafdeling der Wiskunde van de Technische Hogeschool Eindhoven. Deze instellingen, en verder allen die aan de totstandkoming van dit proefschrift hebben meegewerkt, wil ik hier graag bedanken. Mijn moeder dank ik voor het feit dat mijn ouders mij in staat stelden een universitaire opleiding te volgen.

Prof. Dr. F.E.J. Kruzeman Aretz heeft met zijn boeiende colleges mijn belangstelling voor de informatica en de numerieke wiskunde gewekt.

Prof. Dr. T.J. Dekker heeft mij daarna ingewijd op het gebied van de numerieke wiskunde. Dat daarbij mijn weg tot de wetenschap niet langs een militair complex voerde, was een plezierige bijkomstigheid. Voor zijn kritische opmerkingen en zijn bereidheid om mijn copromotor te zijn, ben ik hem zeer erkentelijk.

Dank zij Prof. Dr. P.J. van der Houwen kwam de numerieke integratie van differentiaalvergelijkingen in het middelpunt van mijn belangstelling te staan. In de werkgroep "Beginwaarde problemen" en later in andere werkgroepen, heb ik veel van hem geleerd. Voor zijn stimulerende begeleiding, vooral in de laatste fase toen mijn motivatie niet altijd optimaal was, ben ik hem zeer dankbaar.

Ik stel het op prijs dat Prof. Dr. J.C. Butcher coreferent wil zijn en ik hoop onder zijn leiding in Auckland nuttig onderzoek te kunnen verrichten.

Dr. F.Teer en Dr. S.G. van der Meulen wil ik danken voor de programmatuur die zij ter beschikking hebben gesteld, en D.T. Winter voor de implementatie van TORRIX op de SARA computers.

Met genoegen denk ik terug aan de jaren die ik in Amsterdam heb doorgebracht. De discussies in de diverse werkgroepen van het Mathematisch Centrum waren zeer instruktief. Daarnaast bood de lunchpauze een goede gelegenheid tot ontspanning, waarvoor ik allen die de assistenkamer hebben bewoond, dank. Dik Winter en Joke Blom verdienen hulde voor de wijze waarop zij mij bij problemen met diverse programmatuur geholpen hebben.

Ook in Eindhoven heb ik een jaar prettig gewerkt, waarvoor mijn dank uitgaat naar Prof. Dr. G.W. Veltkamp en A.J. Geurts. De Nederlandse Spoorwegen ben ik erkentelijk voor de vrijwel perfekte service die ik dat jaar mocht genieten.

De Stichting Academisch Rekencentrum Amsterdam heeft met haar rekentuig de verwerking van mijn programma's verzorgd. Haar eindstations vormden aanvankelijk vaak een bron van ergernis, maar later hebben zij mij ook plezierige uurtjes bereid.

De direktie van de Stichting Mathematisch Centrum ben ik bijzonder erkentelijk voor de geboden faciliteiten bij de voorbereiding van mijn vierde artikel, en de dames van de typekamer voor de assistentie bij het typewerk.

Ik dank allen die hebben bijgedragen tot de technische realisatie van dit proefschrift. De typistes van het Mathematisch Centrum onder leiding van Mevr. R.W.T. Riechelmann-Huis, Mej. I. Hoonhout van het Mathematisch Instituut van de Universiteit van Amsterdam en ikzelf hebben het typewerk verzorgd. Margreet Louter-Nool en Joke Blom hebben typefouten opgespoord. De Huisdrukkerij van de Universiteit van Amsterdam verzorgde het drukken.

Tenslotte dank ik Yuri en Mascha voor het niet bevuilen van dit manuscript met hun pootafdrukken, en Selma.

CONTENTS

THE INTRODUCTORY PART

1. INTRODUCTION

The main part of this thesis consists of four papers dealing with various aspects of the numerical integration of partial differential equations. Two of them have been published, and the other ones are submitted for publication in the mathematical literature. These papers are:

[A] *Semi-discretization methods for partial differential equations on non-rectangular grids,* International Journal for Numerical Methods in Engineering 15, 405-419, 1980.

[B] *Generalized Runge-Kutta methods for coupled systems of hyperbolic differential equations,* Journal of Computational and Applied Mathematics 3, 221-233, 1977.

[C] *Formula manipulation in ALGOL 68 and application to Routh's algorithm,* Report 80-01, University of Amsterdam, 1980 (to appear in Computing).

[D] *Stability of linear multistep methods on the imaginary axis,* Report NW 85/80, Mathematisch Centrum, Amsterdam, 1980 (prepublication).

In the first paper we compare several methods for the *semi-discretization* of partial differential equations in two space dimensions. The result of such a semi-discretization is often a system of ordinary differential equations. In the papers [B] and [D] we discuss difference schemes for the numerical solution of these equations, with a special emphasis on the *stability* of these schemes for *hyperbolic* problems. In [B] the main goal is the *construction* of difference schemes, whereas in [D] a stability *analysis* is performed. An *implementation* of important algorithms, occurring in the stability analysis, is given in [C]. These algorithms led to the theoretical results presented in [D].

2. SEMI-DISCRETIZATION OF PARTIAL DIFFERENTIAL EQUATIONS

Many processes in the physical environment give rise to initial boundary value problems; one of the most well-known is the *wave equation*, which is in one-dimensional form described by (RICHTMYER & MORTON [16, page 260])

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \quad c > 0, \quad 0 \leq x \leq 1, \quad t \geq 0.$$

The methods used to solve these problems numerically might be divided into two classes:

A) *Direct grid methods*, such as the ADI-method,

B) Methods using the *method of lines*.

The direct grid methods discretize the variables in both the *space* and the *time* direction. They are usually problem-oriented and in consequence faster than methods belonging to class B. A disadvantage of class A is that each specific method is often only applicable to one special problem, and the analysis is rather cumbersome. The program package TEDDY2 (POLAK [15]) and various methods for the *shallow water equations* (VAN DER HOUWEN [23, section 3.5] , HEAPS [8], KREISS & OLIGER [11], STELLING [20]) belong to this class.

An alternative for these methods is formed by the method of lines (see e.g. BEREZIN & ZHIDKOV [2, section 10.8]). In these methods the space variables are discretized, and the partial differential equation is transformed into a system of ordinary differential equations. The resulting equations are solved numerically with difference schemes for ordinary differential equations. Hence, the original problem is divided into two subproblems, the space-discretization and the time-discretization, and both subproblems may be solved *independent* of each other. The advantages of this approach are clear: Both subproblems allow a thorough analysis, the methods are applicable to wider classes of problems, and an overwhelming amount of class A methods may be produced in this way (n space-discretizers and m time-discretizers yield n times m direct grid methods).

The discretization of the space variables (*semi-discretization*) is

usually based on two methods, either the *finite element* method (STRANG & FIX [21]) or the *finite difference* method (RICHTMYER & MORTON [16]). SCHRYER [18] presents a package which is based on the finite element method, SINCOVEC & MADSEN [19] use a finite difference approximation in their package. Both methods solve initial boundary value problems in one space dimension.

The finite difference approximation for derivatives in one space dimension is usually straightforward. For example, the first derivative $\frac{\partial u}{\partial x}$ may be approximated by standard *central differences*

$$\frac{\partial u}{\partial x} \approx \frac{u(t,x_i+\Delta x) - u(t,x_i-\Delta x)}{2\,\Delta x} ;$$

the error in this approximation is of order $(\Delta x)^2$. One-sided ("up-stream" or "down-stream") differences are sometimes more appropriate with respect to the *stability* of the difference scheme, but their error is of order Δx.

Complications may arise in the construction of difference approximations in two space dimensions. When the domain is well-shaped, e.g. rectangular, their are no special difficulties, but their are important classes of problems with quite irregular domains. We mention once more the shallow water equations (see DRONKERS [5, page 190]):

$$\rho\left[\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - \Omega\, v\right] = -\rho g\frac{\partial h}{\partial x} - \rho g\frac{(u^2+v^2)^{\frac{1}{2}}}{C^2(a_o+h)}\,u ,$$

$$\rho\left[\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \Omega\, u\right] = -\rho g\frac{\partial h}{\partial y} - \rho g\frac{(u^2+v^2)^{\frac{1}{2}}}{C^2(a_o+h)}\,v ,$$

$$\frac{\partial}{\partial x}(a_o+h)\,u + \frac{\partial}{\partial y}(a_o+h)\,v + \frac{\partial h}{\partial t} = 0.$$

These equations describe the motion of long waves in shallow seas, and the boundaries are often irregular. Of course it is possible to cover the domain with a *rectangular* mesh, and approximate the boundaries by straight lines. However, VAN DER HOUWEN [23, section 7.5] showed that it might be profitable to choose a *curvilinear* mesh and adapt the density of the gridlines to the depth of the sea.

For quite another problem, the potential flow past a circle, FREY [6]

demonstrated the usefulness of curvilinear meshes. He presented difference approximations to the space derivatives of first and second order, using function values in nine grid-points. In paper [A] we prove that the error term of these difference approximations is of order $\{\max(\Delta x, \Delta y)\}^2$. Moreover, we propose another difference approximation of the same order, but with much smaller error constants.

3. STABILITY OF DIFFERENCE SCHEMES

The semi-discretization of an initial boundary value problem yields an initial value problem for ordinary differential equations. For example, the standard central difference formula, applied to the wave equation, produces the system (substituting $w=c\frac{\partial u}{\partial x}$ and $v=\frac{\partial u}{\partial t}$)

$$\frac{d\vec{v}}{dt} = c\, J\, \vec{w},$$

$$\frac{d\vec{w}}{dt} = c\, J\, \vec{v},$$

assuming the boundary conditions $v(t,0)=v(t,1)=w(t,0)=w(t,1)=0$. Here, $\vec{v} = [v_1,\ldots,v_n]^T$, and $\vec{w} = [w_1,\ldots,w_n]^T$ are vector functions, giving approximations to the function values $v(t,x)$ and $w(t,x)$ at the grid-points $x_j=j/(n+1)$, and J is the matrix

$$J = \tfrac{1}{2}(\Delta x)^{-1}\begin{pmatrix} 0 & 1 & & & & & \\ -1 & 0 & 1 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & -1 & 0 & 1 \\ & & & & & -1 & 0 \end{pmatrix}.$$

All the eigenvalues of this system are purely *imaginary*, and this means that the exact solution oscillates, just as the solution of the wave equation. Thus, a natural requirement for a difference scheme is that the numerical solution of that scheme remains bounded.

An important class of difference schemes is based on linear multistep methods. The stability requirement leads for these methods to the condition that the roots of the *characteristic polynomial* lie on or within the *unit circle* (see HENRICI [9, section 5.2] , LAMBERT [12]). The unit circle can be mapped onto the left half plane by means of a simple transformation, and the famous algorithms of *Routh* and *Hurwitz* provide methods to determine whether or not all zeros lie in that region (see BARNETT & SILYAK [1], MARDEN [13, Chapter 9], OBRESCHKOFF [14, section 23]). In paper [C] we give an implementation of Routh's algorithm, using polynomial arithmetic only. Moreover, we modified the algorithm in order to avoid an increase in complexity, which is exponential in the degree of the polynomial, without this modification.

4. DIFFERENCE SCHEMES FOR HYPERBOLIC EQUATIONS

The system given in the previous section has eigenvalues lying in a wide range along the imaginary axis. The eigenvalues δ_j are given by

$$\delta_{\pm j} = \pm i \frac{c}{\Delta x} \sin\left(\frac{j\pi}{n+1}\right), \quad j=1,\ldots,n.$$

In many applications one is interested in the long waves, which correspond with the small eigenvalues, whereas the short waves are less relevant. The stability requirements, however, depend on the largest eigenvalues and the time-step prescribed by these requirements is usually much shorter than would be necessary to represent the long waves correctly. Therefore, methods with a large *stability boundary* on the imaginary axis are useful.

The classical fourth order Runge-Kutta method possesses an imaginary stability boundary equal to $2\sqrt{2}$; after division by the number of function evaluations per step we get an *effective stability boundary*, which is approximately equal to 0.71. VAN DER HOUWEN [22, section 2.6.7] presented k-stage Runge-Kutta methods with effective stability boundary equal to 1-1/k, for odd values of k. The order of consistency of these methods is at least two. In paper [B] we try to improve this result by considering generalized Runge-Kutta methods; these methods are suited for the integration of systems like those originating from the wave equations. We

constructed methods of order one, with effective stability boundaries equal to 2-2/k, if k is odd, and methods of order two, with effective stability boundary 5/3. SCHIPPER [17] applied these methods to a semi-discretized second order differential equation, describing the displacement of a cable, which is chained to the sea bottom on one side:

$$50\,\frac{\partial^2 y}{\partial t^2} + c\,\frac{\partial y}{\partial t} = -\,0.139\;10^7\,\frac{\partial^4 y}{\partial x^4} + 10^5\,\frac{\partial^2 y}{\partial x^2} + h(y),\quad 0\le x\le 200,\quad t\ge 0.$$

Other numerical results are given in (DEKKER [4]).

A disadvantage of *explicit* methods, such as the before mentioned Runge-Kutta methods, is the severe restriction on the integration step, imposed by the stability condition. *Implicit* methods, like linear multistep methods, may be more attractive. The trapezoidal rule, for example, is A-stable (DAHLQUIST [3]), and hence stable along the imaginary axis (JELTSCH [10]). However, higher order linear multistep methods behave poor along the imaginary axis; the well-known backward differentiation formulas (GEAR [7]) of order three and higher have very small imaginary stability boundaries. In paper [D] we prove in fact that the imaginary stability boundary is a function of the error constant of the third order term. For any method of order higher than two, the stability boundary is at most $\sqrt{3}$. The value $\sqrt{3}$ is obtained by the fourth order Milne-Simpson method.

REFERENCES

[1] BARNETT, S. & D.D. SILYAK, *Routh's algorithm: a centennial survey*, SIAM Review 19, 472-489, 1977.

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

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

[4] 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.

[5] DRONKERS, J.J., *Tidal computations in rivers and coastal waters*, North-Holland publishing company, Amsterdam, 1964.

[6] FREY, W.H., *Flexible finite difference stencils from isoparametric finite elements*, Int. J. num. Meth. Engng. 11, 1653-1665, 1977.

[7] GEAR, C.W., *Numerical initial value problems in ordinary differential equations*, Prentice Hall, Englewood Cliffs, N.J., 1971.

[8] HEAPS, N.S., *A two-dimensional numerical sea model*, University of Liverpool, 1969.

[9] HENRICI, P., *Discrete variable methods in ordinary differential equations*, John Wiley & Sons, New York, 1962.

[10] JELTSCH, R., *Stability on the imaginary axis and A-stability of linear multistep methods*, BIT 18, 170-174, 1978.

[11] KREISS, H. & J. OLIGER, *Methods for the approximate solution of time dependent problems*, GARP publication series, no 10, 1973.

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

[13] MARDEN, M., *Geometry of polynomials*, Amer. Math. Soc., Providence, 1966.

[14] OBRESCHKOFF, N., *Verteilung und Berechnung der Nullstellen reeler Polynome*, VEB Deutscher Verlag der Wissenschaften, Berlin, 1963.

[15] POLAK, S.J. & J.SCHROOTEN, *Preliminary TEDDY2 user manual*, Philips-ISA-UDV-SCA/SP/75/024/mw, Eindhoven, 1975.

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

[17] SCHIPPER, A., *Experiments with a generalized Runge-Kutta method by solving a second order differential equation*, master thesis, part 1, University of Amsterdam, 1978.

[18] SCHRYER, N.L., *Numerical solution of time-varying partial differential equations in one space variable*, Comp. Science Techn. Report No 53, Bell Laboratories, Murray Hill, 1974.

[19] SINCOVEC, R.F. & N.K. MADSEN, *Software for nonlinear partial differential equations*, ACM Transactions on Mathematical Software 1, 232-260, 1975.

[20] STELLING, G.S., *The stability of the leap frog scheme and a proposal for the improvement of the leap frog scheme for the shallow water equations in one dimension*, Report S 333, Waterloopkundig Laboratorium, Delft, 1977.

[21] STRANG, G. & G.J. FIX, *An analysis of the finite element method*, Prentice Hall, Englewood Cliffs, N.J., 1973.

[22] VAN DER HOUWEN, P.J., *Construction of integration formulas for initial value problems*, North-Holland publishing company, Amsterdam, 1976.

[23] VAN DER HOUWEN, P.J., *Berekening van de waterbeweging in zeeën en rivieren*, MC syllabus 33, Mathematisch Centrum, Amsterdam, 1977.

THE FOUR PAPERS

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 15, 405–419 (1980)

SEMI-DISCRETIZATION METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS ON NON-RECTANGULAR GRIDS

K. DEKKER

Mathematical Centre, Amsterdam, The Netherlands

INTRODUCTION

The problem of approximating the solution of time-dependent partial differential equations (PDEs) is often solved by using direct grid methods, e.g. the alternating direction, the locally one-dimensional, and the hopscotch methods.[2,10] An alternative approach consists in splitting the problem into two subproblems: first, transforming the PDE into a system of ordinary differential equations (ODEs) by discretizing the space variables, and secondly, solving the resulting system of ODEs with a suitable integrator. At the moment several packages based on this idea are available.[6,7] At the Mathematical Centre some investigation is done in this area, too. Several generalized splitting methods have been constructed,[3] which can integrate ODEs with five- and nine-point coupling, thereby setting a need for semi-discretization methods which yield a nine-point coupling.

On a square mesh, the approximation of the first and second derivatives of the dependent variable may be obvious. However, on non-rectangular grids it is not at all clear which finite difference formula is the best. Therefore, we will compare three semi-discretization methods in this paper. The first one is a special case of a method published recently by Frey,[1] for which we give an alternative formulation admitting strict error bounds. The second method is developed by Kok *et al.*,[4] originally intended for grids with an explicitly given mesh, and the third one is a new method, which tries to minimize the truncation error of the derivative-approximations.

In the next sections we will derive and summarize the formula on which the three methods are based. Furthermore, we calculate error bounds for Frey's method.

In the last section we give some numerical results of the three methods. First we compute the error terms for a variety of different grids, then we calculate the errors in the derivatives of a set of analytically given functions on a fixed grid, and finally we compute the solution of an elliptic PDE using the three methods.

THE TRANSFORMATION METHOD OF FREY

Let a curvilinear grid $R \subset \mathbb{R}^2$ be given, together with the function values of a sufficiently differentiable function $u(x, y)$ at the grid points. The problem of approximating the first and second derivatives of u in an interior grid point, can be solved by considering a transformation T from an element E of R to a square element E' (Figure 1). This method was proposed by Frey; we will formulate it now in a slightly different notation.

As T is an only locally defined transformation, we may assume without loss of generality that z_5 and w_5 are the origins of their respective planes. Then, we define the grid distance Δ and the element E' as follows.

0029–5981/80/0315–0405$01.00

Received 2 August 1978
Revised 2 April 1979

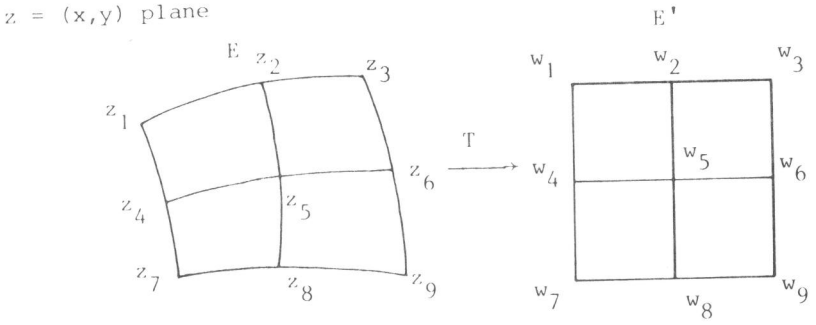


Figure 1. Curvilinear and square element

Definition 1

$$\Delta = \max_{i=1,\ldots,9} \|z_i\|_1; \tag{1}$$

here $\| \ \|_1$ denotes the maximum norm, and z denotes the vector $(x, y)^T$. The grid points of E' are given by

$$w_{i-3j+5} = \begin{pmatrix} i & \Delta \\ j & \Delta \end{pmatrix}; \qquad i = -1, 0, 1; j = -1, 0, 1 \tag{2}$$

Now, we can express the transformation T as a Taylor series about z_5:

$$T(z) = T(z_5) + T'(z_5)(z - z_5) + \tfrac{1}{2}T''(z_5)(z - z_5)^2 + 0(z - z_5)^3 \tag{3a}$$

In order to simplify the notation we will rename the first and second derivatives of T at z_5 by the linear operator A and the bilinear operator B. Recalling that we assume z_5 and $T(z_5)$ to be zero-vectors, we may rewrite (3a) as

$$T(z) = Az + \tfrac{1}{2}Bz^2 + 0(z^3) \tag{3b}$$

When the operators A and B are known, we can express the derivatives of u in z_5 by (using the chain rule for differentiation)

$$\begin{Bmatrix} u_x \\ u_y \end{Bmatrix} = A^T \begin{Bmatrix} u_X \\ u_Y \end{Bmatrix} \tag{4}$$

$$\begin{bmatrix} u_{xx} & u_{xy} \\ u_{yx} & u_{yy} \end{bmatrix} = A^T \begin{bmatrix} u_{XX} & u_{XY} \\ u_{YX} & u_{YY} \end{bmatrix} A + \langle u_X u_Y \rangle B \tag{5}$$

Here, we assume u to be the function on E' defined by $u(w) = u(z)$ if $w = Tz$, and using central difference formulae on E', we can calculate the approximations for u_x, u_y, u_{xx}, u_{xy} and u_{yy} on E.

In order to compute approximations to A and B, we consider an operator $\mathbf{S}: E' \to E$, such that $\mathbf{S}(w_i) = z_i$ for $i = 2, 4, 5, 6, 8$, and $\mathbf{S}(w_i) = z_i + 0(\Delta^3)$ for the other gridpoints. It is easily verified that these conditions are satisfied if $\mathbf{S}$ is given by

$$\mathbf{S}(w) = \mathbf{C}w + \tfrac{1}{2}Dw^2, \tag{6}$$

with

$$\mathbf{C} = \frac{1}{2\Delta}\begin{bmatrix} x_6 - x_4 & x_2 - x_8 \\ y_6 - y_4 & y_2 - y_8 \end{bmatrix} \tag{7}$$

$$\mathbf{D} = \frac{1}{\Delta^2}\begin{bmatrix} x_6 + x_4 & \dfrac{-x_1 + x_3 + x_7 - x_9}{4} & \dfrac{-x_1 + x_3 + x_7 - x_9}{4} & x_2 + x_8 \\ y_6 + y_4 & \dfrac{-y_1 + y_3 + y_7 - y_9}{4} & \dfrac{-y_1 + y_3 + y_7 - y_9}{4} & y_2 + y_8 \end{bmatrix} \tag{8}$$

Now, recalling that $T(z_i) = w_i$, we see that $\mathbf{S}$ is an approximation to T^{-1}, so that we are able to compute approximations $\mathbf{A}$ and $\mathbf{B}$ to A and B, using

$$z = \mathbf{S}(\mathbf{A}z + \tfrac{1}{2}\mathbf{B}z^2 + 0(z^3)) \tag{9}$$

Comparing terms of the same order in (9), we finally find

$$\mathbf{A} = \mathbf{C}^{-1} \tag{10}$$

and

$$\mathbf{B} = -\mathbf{ADAA} \tag{11}$$

Substituting $\mathbf{A}$ for A and $\mathbf{B}$ for B in (4) and (5), we obtain approximations $\mathbf{u}_x$ to u_x, etc., which are exactly the same as equations (11)–(14) given by Frey, as some lengthy calculations may reveal. However, the above formulation was chosen because it enables us to derive error bounds for the approximations.

In order to perform the error analysis we first notice that the inverse of T can be written as (regarding x and y as functions of X and Y)

$$\begin{aligned} T^{-1}(w) = S(w) &= \begin{bmatrix} x_X & x_Y \\ y_X & y_Y \end{bmatrix} w + \tfrac{1}{2}\begin{bmatrix} x_{XX} & x_{XY} & x_{YX} & x_{YY} \\ y_{XX} & y_{XY} & y_{YX} & y_{YY} \end{bmatrix} w^2 + 0(w^3) \\ &= Cw + \tfrac{1}{2}Dw^2 + 0(w^3) \end{aligned} \tag{12}$$

Expanding x and y in a Taylor series about w_5, we obtain (for small Δ)

$$\delta C = \mathbf{C} - C = \frac{\Delta^2}{6}\begin{bmatrix} x_{XXX} & x_{YYY} \\ y_{XXX} & y_{YYY} \end{bmatrix} + 0(\Delta^4) \tag{13}$$

$$\delta D = \mathbf{D} - D = \frac{\Delta^2}{12}\begin{bmatrix} x_{XXXX} & 2(x_{XXXY} + x_{XYYY}) & 2(x_{XXXY} + x_{XYYY}) & x_{YYYY} \\ y_{XXXX} & 2(y_{XXXY} + y_{XYYY}) & 2(y_{XXXY} + y_{XYYY}) & y_{YYYY} \end{bmatrix} + 0(\Delta^4)$$

Hence, we have $\|\delta C\| \le k_c\Delta^2$ and $\|\delta D\| \le k_d\Delta^2$. Using $A = C^{-1}$, $B = -ADAA$, we find after some calculation (cf. Wilkinson[8])

$$\delta A = \mathbf{A} - A = -\mathbf{A}\delta C(I - \mathbf{A}\delta C)^{-1}\mathbf{A}, \text{ or } \|\delta A\|_2 \le \frac{\|\mathbf{A}\|_2^2}{1 - \|\mathbf{A}\|_2 k_c \Delta^2} k_c\Delta^2 = k_a\Delta^2 \tag{14}$$

$$\|\delta B\|_2 = \|\mathbf{B} - B\|_2 \le \|\mathbf{A}\|_2^2\left\{\frac{3k_a}{\{1 - \|A\|_2 k_c\Delta^2\}^2}\|\mathbf{D}\|_2 + \|\mathbf{A}\|_2 k_d\right\}\Delta^2 = k_b\Delta^2 \tag{15}$$

Obviously, the errors δC and δD depend on the curvature of the gridlines, and the variation in the distance between the gridpoints; δA and δB are influenced, too, by the condition of the transformation, and in consequence by the angle between the gridlines.

Now, let $\mathbf{u}_X$, etc., denote the central difference approximation to u_X; using the error bounds given above, and formulae (4), (5), we obtain the error estimates

$$\left\| \begin{Bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \end{Bmatrix} - \begin{Bmatrix} u_x \\ u_y \end{Bmatrix} \right\|_2 \leq \left\{ k_a \left\| \begin{Bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \end{Bmatrix} \right\|_2 + \tfrac{1}{6} \|\mathbf{A}\|_2 \left\| \begin{matrix} u_{XXX}(\xi) \\ u_{YYY}(\eta) \end{matrix} \right\|_2 \right\} \Delta^2 \tag{16}$$

where ξ and η are certain points between w_4 and w_6, w_2 and w_7, respectively. In order to compute the error in the second derivative, we observe that

$$\mathbf{S}w_i = z_i + \delta z_i, \qquad i = 1, 3, 7, 9, \tag{17}$$

with e.g.

$$\delta z_3 = \langle -\tfrac{1}{2}x_{XXY} - \tfrac{1}{2}x_{XYY},\ -\tfrac{1}{2}y_{XXY} - \tfrac{1}{2}y_{XYY} \rangle^{\mathrm{T}} \Delta^3 + 0(\Delta^4)$$

and

$$\sum_i \delta z_i = 0(\Delta^4)$$

Thus, the error in $u_{XY} - \mathbf{u}_{XY}$ is determined not only by the discretization but also by the evaluation of u in the points z_i instead of $\mathbf{S}w_i$. Using $\mathbf{u}_{XY} = 1/4\Delta^2\{u_3 + u_7 - u_1 - u_9\}$, this last error is given by

$$\left| \frac{1}{4\Delta^2} \left\{ \langle u_x u_y \rangle \langle \delta z_3 + \delta z_7 - \delta z_1 - \delta z_9 \rangle + \begin{bmatrix} u_{xx} & u_{xy} \\ u_{yx} & u_{yy} \end{bmatrix} 0(\Delta^4) \right\} \right| = c\Delta^2 \tag{18}$$

because $\delta z_3 + \delta z_7 - \delta z_1 - \delta z_9 = 0$. Finally we get

$$\left\| \begin{bmatrix} \mathbf{u}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx} & \mathbf{u}_{yy} \end{bmatrix} - \begin{bmatrix} u_{xx} & u_{xy} \\ u_{yx} & u_{yy} \end{bmatrix} \right\|_2 \leq \left[\|A\|_2^2 \left\{ \tfrac{1}{12} \left\| \begin{bmatrix} u_{XXXX} & 2(u_{XXXY} + u_{XYYY}) \\ 2(u_{XXXY} + u_{XYYY}) & u_{YYYY} \end{bmatrix} \right\|_2 + c \right\} \right.$$

$$\left. + 2 \left\| \begin{bmatrix} \mathbf{u}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx} & \mathbf{u}_{yy} \end{bmatrix} \right\|_2 \|A\|_2 k_a + \|\mathbf{B}\|_2 \left\| \begin{matrix} u_{XXX} \\ u_{YYY} \end{matrix} \right\|_2 + \left\| \begin{matrix} \mathbf{u}_X \\ \mathbf{u}_Y \end{matrix} \right\|_2 k_b \right] \Delta^2 \tag{19}$$

The estimates (16) and (19) show that the finite difference approximations are of second order, just as is the case for square elements. However, for curvilinear elements the error constants will be larger, as more terms occur in the error bounds.

A TRANSFORMATION METHOD WITH EXPLICITLY GIVEN GRIDLINES

The method of Kok *et al.*[4] is based on the idea that the gridlines are known functions of the co-ordinates z and y, such that their derivatives can be obtained easily. When the gridlines are not known, they are locally approximated; to that end the lines in the x-direction are considered to be functions of x, and a three-point interpolation formula yields

$$y = f_-(x) = y_7 + \frac{y_8 - y_7}{x_8 - x_7}(x - x_7) + \frac{\dfrac{y_9 - y_8}{x_9 - x_8} - \dfrac{y_8 - y_7}{x_8 - x_7}}{x_9 - x_7}(x - x_7)(x - x_8) \tag{20}$$

and similarly for f_+, f, g_+, g and g_- (Figure 2).

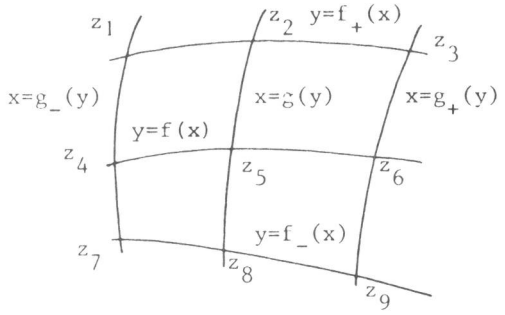


Figure 2. The gridlines as second-order interpolation polynomials

Now, we will denote by f, f_+, f', etc., the function and derivative values at the point x_5, and similarly by g, g_+ the values at the point y_5. Using the same notation as in the previous section, the derivatives of a function u at the point $z_5 = (x_5, y_5)$ are given by formulae (4) and (5), with matrix A defined by

$$\begin{aligned} A_{11} &= \frac{2\Delta}{g_+ - g}, & A_{22} &= \frac{2\Delta}{f_+ - f_-} \\ A_{12} &= -g' A_{11}, & A_{21} &= -f' A_{22} \end{aligned} \tag{21}$$

and B defined by

$$\begin{aligned} B_{111} &= -8\frac{g_+ - 2g + g_-}{(g_+ - g_-)^3}\Delta, & B_{222} &= -8\frac{f_+ - 2f + f_-}{(f_+ - f_-)^3}\Delta \\ B_{112} &= B_{121} = B_{111}\frac{A_{12}}{A_{11}} - A_{11}^2\frac{g'_+ - g'_-}{2\Delta}, & B_{212} &= B_{221} = B_{222}\frac{A_{21}}{A_{22}} - A_{22}\frac{f'_+ - f'_-}{2\Delta} \\ B_{122} &= -\{(g')^2 B_{111} + 2g' B_{112} + g'' A_{11}\}, & B_{211} &= -\{(f')^2 B_{222} + 2f' B_{221} + f'' A_{22}\} \end{aligned} \tag{22}$$

For a detailed derivation of these formulae we refer to Kok *et al.*[4]

THE MINIMIZATION METHOD

In the previous sections we have described two discretization methods based on a transformation of the elements. Here, we will follow an alternative approach. An approximation formula might be regarded as a function of nine parameters, the weights in the points z_i, $i = 1, \ldots, 9$. Expanding each function value in z_i as a Taylor series, we obtain for the approximation formula a Taylor series about z_5. Each term of this series has a coefficient depending on several of the weights used in the formula. Now, we may wish these coefficients to have a predetermined value, for example one for the derivative to be approximated, and zero for the other coefficients up to and including third order. However, we obtain ten equations with nine unknowns in this way, and a solution generally does not exist. (Here, we note that in the rectangular case, a (not unique) solution exists). Restricting ourselves to the lower order terms, we get only six equations, and we have a wide variety of solutions to them. In the following we will compute that solution to the latter system, which minimizes the error constants of the

third-order terms, and hope that this solution yields a good approximation to the required derivative.

First, we introduce the following notations:

$$
\begin{aligned}
u^{\mathrm{T}} &= \langle u_1 - u_5, u_2 - u_5, \ldots, u_4 - u_5, u_6 - u_5, \ldots, u_9 - u_5 \rangle \\
d^{\mathrm{T}} &= \left\langle u_x, u_y, \frac{u_{xx}}{\sqrt{2}}, u_{xy}, \frac{u_{yy}}{\sqrt{2}}, \frac{u_{xxx}}{\sqrt{6}}, \frac{u_{xxy}}{\sqrt{2}}, \frac{u_{xyy}}{\sqrt{2}}, \frac{u_{yyy}}{\sqrt{6}} \right\rangle \\
\mathbf{d}^{\mathrm{T}} &= \left\langle \mathbf{u}_x, \mathbf{u}_y, \frac{\mathbf{u}_{xx}}{\sqrt{2}}, \mathbf{u}_{xy}, \frac{\mathbf{u}_{yy}}{\sqrt{2}} \right\rangle \\
d_s^{\mathrm{T}} &= \left\langle u_x, u_y, \frac{u_{xx}}{\sqrt{2}}, u_{xy}, \frac{u_{yy}}{\sqrt{2}} \right\rangle
\end{aligned}
\tag{23}
$$

Here, u_i denotes the function value in z_i (cf. Figure 1), u_x, etc., the derivatives in z_5, and $\mathbf{u}_x$, etc., approximations to these derivatives. Further, we note that we have added the factors $\sqrt{2}$ and $\sqrt{6}$, in order to have a rotation independent Euclidean norm $\| \ \|_{\mathrm{E}}$. This is illustrated by the following example.

Example 1

Consider the function $u(x, y) = x^2 - y^2$. The second derivatives are $\langle u_{xx}, u_{xy}, u_{yx}, u_{yy} \rangle = \langle 2, 0, 0, -2 \rangle$ and their Euclidean norm is $2\sqrt{2}$. A rotation of $\pi/4$ yields the function $u(\xi, \eta) = 2\xi\eta$, with second derivatives given by $\langle 0, 2, 2, 0 \rangle$, again with Euclidean norm $2\sqrt{2}$. As u_{xy} equals u_{yx}, the latter term can be deleted to shorten the notation. However, the norm is influenced by this deletion and $\|\langle u_{xx}, u_{xy}, u_{yy} \rangle\|_{\mathrm{E}}$ is no longer rotation independent. Thus, we have to divide u_{xx} and u_{yy} by a factor $\sqrt{2}$ in order to preserve this property.

Now, a nine-point approximation method is defined by a matrix of weights $W(5 \times 8)$

$$\mathbf{d} = Wu \tag{24}$$

Furthermore, expanding u_i in a Taylor series about z_5, we have

$$u = Md + 0(\Delta^4), \tag{25}$$

where M is a 8×9 matrix, only depending on the size and shape of the element E (see Figure 1). The error in the approximation can be expressed by

$$\mathbf{d} - d_s = WMd + \text{higher order terms} - d_s \tag{26}$$

and we call F defined by

$$F_{ij} = (WM)_{ij} - \delta_{ij}, \qquad i = 1, \ldots, 5, \qquad j = 1, \ldots, 9 \tag{27}$$

the error matrix. Obviously, F depends only on the given element, and the approximation method chosen, and the minimizing problem is nothing else than constructing a kind of inverse

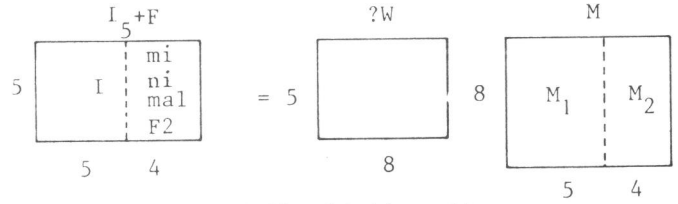


Figure 3. The minimizing problem

of M. To be more precise, we want to minimize $\|F\|_E$ under the constraints $F_{ij} = 0$, $i, j = 1, \ldots, 5$ (these are the first- and second-order terms), as is illustrated in Figure 3.

The solution may be found by numerical algebra techniques. First, we compute the pseudo-inverse M_1^+ of M_1 (Wilkinson and Reinsch[9]), such that $M_1^+ M_1 = I$. Now, let $M_1^\perp$ be the matrix consisting of the three vectors which are orthogonal to M_1. Then we solve the overdetermined systems

$$(M_2^T M_1^\perp) X = (M_1^\perp M_2)^T, \qquad X \text{ a } 3 \times 5 \text{ matrix} \tag{28}$$

in the least squares sense, and the desired optimal solution W_{opt} is given by

$$W_{opt} = M_1^+ - (M_1^\perp X)^T \tag{29}$$

It is easily verified that this solution is optimal, by considering

$$F_2 = (M_1^+ - (M_1^\perp X)^T) M_2 \tag{30}$$

In the next sections, the computations of W_{opt} were made by using the NAG-library[5] routine F01BHF, which performs singular value decomposition.

Remarks

(1) We note that the matrix M_1 has rank 5, if and only if no quadratic form exists, which is satisfied by the nine points z_i. As a quadratic form is defined by five points, M_1 will always have full rank, unless the grid is chosen very awkwardly (e.g. all points on a circle, on two straight lines). Thus M_1^+ indeed exists.

(2) It may be advisable to scale the matrix M before executing the formulas (28) and (29), in order to avoid ill-conditioning. Division of the first two columns of M by Δ, the next three by Δ^2 and the last four by Δ^3 (Δ as defined in (1)) will be appropriate. Afterwards, the first two rows of W_{opt} should be divided by Δ, the next ones by Δ^2.

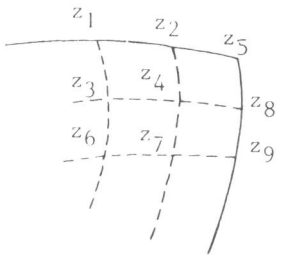


Figure 4. An element on the boundary

(3) In the derivation of W_{opt}, we did not use any specific property of the nine points given, except that they did not lie on a quadratic form. Thus, the method can be used for boundary points, too, when we select 8 points in the neighbourhood of the boundary point, all lying within the domain or on the boundary (Figure 4). (Note that the numbering is irrelevant.) However, as the distances $z_i - z_5$ are larger for boundary points than for interior points, the approximations will be less accurate.

(4) It may happen that the matrix $(M_2^T M_1^\perp)$ is not of full rank. This case occurs if the three-dimensional space $M_1^\perp$ is not spanned by the columns of M_2, or equivalently, if the nine columns of M do not span the whole space $\mathbb{R}^8$. For example, when the element is uniform and rectangular, M spans only a 7-dimensional subspace of $\mathbb{R}^8$. When the above-described situation

arises, the solution of equation (28) is not unique; using the pseudo-inverse of $(M_2^{\mathrm{T}}M_1^{\perp})$, we obtain the unique solution vector with minimal norm in this solution space, for each of the five columns of the right-hand side. However, on square elements we do not obtain the usual central difference formulae by this method, although these formulae obviously lie in the solution space of (28). The difficulty was overcome by expanding the matrix M with five columns, whose elements were given by $x_i^4/\sqrt{24}$, $x_i^3y_i/\sqrt{6}$, $x_i^2y_i^3/2$, $x_iy_i^3/\sqrt{6}$, $y_i^4/\sqrt{24}$; all divided by the scaling factor $10\Delta^4$.

(5) Finally, we remark that the scheme can be adapted to boundary conditions other than those of Dirichlet type. To that end we have to redefine the vector u by replacing the element $u_i - u_5$ by the value of the boundary condition at the point z_i. Again, expansion of u in a Taylor series about z_5, equation (25), yields a matrix M, and equations (28) and (29) are applied to find the new weights. For example, let the boundary condition at z_i be given by

$$b_i = b(z_i) = \alpha u_i + \beta u_{x,i} + \gamma u_{y,i}$$

Expanding $b_i - \alpha u_5$ yields

$$b_i - \alpha u_5 = m_i^{\mathrm{T}} d + 0(\Delta^4) \tag{25a}$$

so we replace the ith row of M by m_i^{T}. Of course, the least squares problem to be solved is linear, only if the boundary conditons are linear, too.

NUMERICAL EXAMPLES

In this section we will give some numerical results, produced by the methods described in the previous sections. These methods will be denoted by their generating weight-matrices W_2 (Frey's method), W_3 (Kok's method) and W_4 (the minimization method).

In the first subsection we will compute the entries of the error matrix F defined by (27) for various elements. Here, it turns out that W_2, W_3 and W_4 produce identical error matrices on rectangular elements, whereas the F generated by W_4 has the smallest norm on curvilinear elements.

In the next subsection we approximate the derivatives of a set of analytic functions with the various methods on several non-rectangular grids. Again, W_4 gives the best results, as it shows the highest order of convergence when the functions are smoothed.

Finally, we solved an elliptic problem on a curvilinear grid, and tabulated the error between the analytic solution and the solution of the discrete system for various numbers of gridlines.

The error matrices F for the methods W_2, W_3 and W_4

In this series of tests we computed the entries of F for elements given by the formulae (see also Figure 5)

$$\begin{Bmatrix} x_6 \\ y_6 \end{Bmatrix}, \begin{Bmatrix} x_4 \\ y_4 \end{Bmatrix} = \mathrm{d}x \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{Bmatrix} 1 + c_1\,\mathrm{d}x \\ c_2\,\mathrm{d}x \end{Bmatrix}, \begin{Bmatrix} -1 + c_1\,\mathrm{d}x \\ c_2\,\mathrm{d}x \end{Bmatrix}$$

$$\begin{Bmatrix} x_2 \\ y_2 \end{Bmatrix}, \begin{Bmatrix} x_8 \\ y_8 \end{Bmatrix} = \mathrm{d}y \begin{bmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) \end{bmatrix} \begin{Bmatrix} 1 + c_3\,\mathrm{d}y \\ c_4\,\mathrm{d}y \end{Bmatrix}, \begin{Bmatrix} 1 + c_3\,\mathrm{d}y \\ c_4\,\mathrm{d}y \end{Bmatrix} \tag{31}$$

$$z_1 = z_2 + z_4 + c_5\,\mathrm{d}x\,\mathrm{d}y \begin{Bmatrix} \mathrm{d}x \\ \mathrm{d}y \end{Bmatrix}, \qquad z_3 = z_2 + z_6 + c_5\,\mathrm{d}x\,\mathrm{d}y \begin{Bmatrix} \mathrm{d}x \\ -\mathrm{d}y \end{Bmatrix}$$

$$z_7 = z_4 + z_8 + c_5\,\mathrm{d}x\,\mathrm{d}y \begin{Bmatrix} -\mathrm{d}x \\ \mathrm{d}y \end{Bmatrix}, \qquad z_9 = z_6 + z_8 + c_5\,\mathrm{d}x\,\mathrm{d}y \begin{Bmatrix} -\mathrm{d}x \\ -\mathrm{d}y \end{Bmatrix}$$

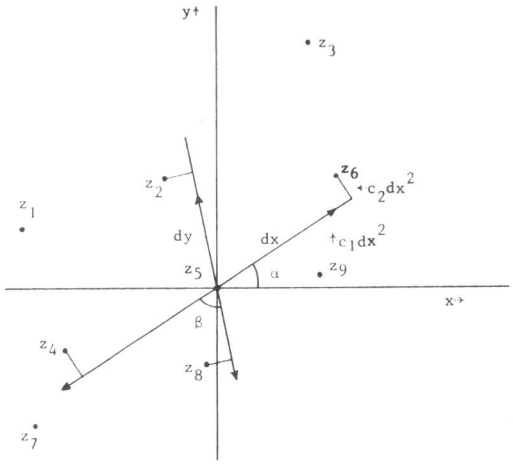


Figure 5. The element defined by formulae (31)

For the parameters in formulae (31) we have chosen the default values

$$\begin{aligned} \alpha &= 0 \\ \beta &= \pi/2 \\ \mathrm{d}x &= \mathrm{d}y = 1 \\ c_1 &= c_2 = c_3 = c_4 = c_5 = 0 \end{aligned} \tag{32}$$

and in each test we varied some of these parameters, namely

(a) $\beta = \dfrac{i\pi}{18}$, $i = 1, \ldots, 9$, yielding a diamond

(b) $\mathrm{d}y = \dfrac{i}{10}$, $i = 1, \ldots, 10$, yielding a rectangle

(c) $\mathrm{d}y = \dfrac{i}{10}$, $\beta = \dfrac{10+i}{40}\pi$, $i = 1, \ldots, 10$, yielding a parallelogram

(d) $c_5 = \dfrac{i}{20}$, $i = 1, \ldots, 10$, yielding a distorted square (33)

(e) $c_1 = c_2 = c_3 = c_4 = \dfrac{i}{20}$, $i = 1, \ldots, 10$, yielding a curvilinear element, in which each quadrilateral remains a parallelogram

(f) $\mathrm{d}y = 0{\cdot}5$, $\beta = \dfrac{\pi}{3}$, $c_1 = \dfrac{i}{30}$, $c_2 = -\dfrac{i}{20}$, $c_3 = \dfrac{i}{50}$, $c_4 = \dfrac{i}{40}$, $c_5 = \dfrac{i}{40}$
$i = 1, \ldots, 10$, yielding a curvilinear element

In Tables I–VI we listed the non-zero values of $f_{11}, f_{12}, f_{13}, f_{21}, f_{22}$ and f_{23} which are defined by the matrix F in the following way:

$$F \text{ a } 5\times 9 \text{ matrix}$$

$$\begin{array}{c} 2 \\ 3 \end{array}\left[\begin{array}{c|c|c} F_{11} & F_{12} & F_{13} \\ \hline F_{21} & F_{22} & F_{23} \end{array}\right], \qquad f_{ij} = \|F_{ij}\|_{\mathrm{E}} \tag{34}$$

$$\begin{array}{ccc} 2 & 3 & 4 \end{array}$$

As the values of f_{ij} turned out to be independent of α, we dropped this parameter from the tests.

Table I. f_{13} element for (33a)

β	W_2	W_3	W_4
$\pi/18$	0·810	0·810	0·014
$2\pi/18$	0·792	0·792	0·057
$3\pi/18$	0·764	0·764	0·126
$4\pi/18$	0·727	0·727	0·216
$5\pi/18$	0·686	0·686	0·320
$6\pi/18$	0·646	0·646	0·423
$7\pi/18$	0·610	0·610	0·507
$8\pi/18$	0·586	0·586	0·560
$\pi/2$	0·577	0·577	0·577

Table II. f_{13} element (33b)

dy	W_2, W_3, W_4
0·1	0·408
0·2	0·409
0·3	0·410
0·4	0·413
0·5	0·421
0·6	0·434
0·7	0·455
0·8	0·485
0·9	0·525
1·0	0·577

Table III. f_{13} for element (33c)

dy	β	W_2	W_3	W_4
0·1	$11\pi/40$	0·536	0·536	0·245
0·2	$12\pi/40$	0·503	0·503	0·272
0·3	$13\pi/40$	0·478	0·478	0·299
0·4	$14\pi/40$	0·461	0·461	0·327
0·5	$15\pi/40$	0·453	0·453	0·358
0·6	$16\pi/40$	0·455	0·455	0·391
0·7	$17\pi/40$	0·467	0·467	0·429
0·8	$18\pi/40$	0·491	0·491	0·473
0·9	$19\pi/40$	0·527	0·527	0·522
1·0	$\pi/2$	0·577	0·577	0·577

Table IV. f_{13} and f_{22} for element (33d)

	f_{13}			f_{22}		
c_5	W_2	W_3	W_4	W_2	W_3	W_4
0·05	0·577	0·577	0·576	0·100	0·100	0
0·1	0·577	0·577	0·573	0·200	0·200	0
0·15	0·577	0·577	0·568	0·301	0·301	0
0·2	0·577	0·577	0·560	0·402	0·402	0
0·25	0·577	0·577	0·549	0·504	0·405	0
0·3	0·577	0·577	0·533	0·607	0·607	0
0·35	0·577	0·577	0·513	0·711	0·711	0
0·4	0·577	0·577	0·488	0·816	0·816	0
0·45	0·577	0·577	0·458	0·922	0·922	0
0·5	0·577	0·577	0·423	1·031	1·031	0

Table V. f_{12}, f_{13}, f_{22} and f_{23} for element (33e)

	f_{12}			f_{13}			f_{22}			f_{23}		
c_1	W_2	W_3	W_4	W_2	W_3	W_4	W_2	W_3	W_4	W_2	W_3	W_4
0·05	0·122	0·122	0	0·582	0·582	0·417	0·010	0·017	0	0·182	0·183	0·066
0.1	0·245	0·245	0	0·595	0·595	0·405	0·040	0·070	0	0·364	0·365	0·128
0·15	0·367	0·368	0	0·618	0·619	0·399	0·090	0·159	0	0·543	0·548	0·191
0·2	0·490	0·492	0	0·653	0·655	0·392	0·160	0·286	0	0·719	0·734	0·252
0·25	0·612	0·618	0	0·700	0·706	0·383	0·250	0·458	0	0·893	0·931	0·310
0·3	0·735	0·749	0	0·760	0·775	0·371	0·360	0·683	0	1·063	1·151	0·364
0·35	0·857	0·890	0	0·836	0·868	0·356	0·490	0·977	0	1·23	1·42	0·414
0·4	0·980	1·05	0	0·926	0·991	0·337	0·640	1·37	0	1·40	1·77	0·459
0·45	1·10	1·24	0	1·03	1·16	0·312	0·810	1·92	0	1·56	2·28	0·497
0·5	1·22	1·47	0	1·15	1·39	0·280	1·00	2·74	0	1·73	3·09	0·530

Table VI. f_{12}, f_{13}, f_{21} and f_{23} for element (33f)

	f_{12}			f_{13}			f_{21}			f_{22}			f_{23}		
c_2	W_2	W_3	W_4	W_2	W_3	W_4	W_2	W_3	W_4	W_2	W_3	W_4	W_2	W_3	W_4
−0·05	0·081	0·081	0	0·480	0·480	0·284	0	0·004	0	0·056	0·056	0	0·115	0·115	0·058
−0·1	0·163	0·163	0	0·485	0·485	0·270	0	0·018	0	0·118	0·126	0	0·228	0·229	0·108
−0·15	0·244	0·245	0	0·493	0·494	0·261	0	0·043	0	0·187	0·219	0	0·339	0·344	0·162
−0·2	0·326	0·327	0	0·506	0·507	0·252	0	0·086	0	0·265	0·340	0	0·449	0·462	0·217
−0·25	0·407	0·410	0	0·523	0·525	0·241	0	0·156	0	0·354	0·495	0	0·557	0·586	0·273
−0·3	0·488	0·494	0	0·547	0·548	0·229	0	0·268	0	0·454	0·692	0	0·664	0·720	0·329
−0·35	0·570	0·578	0	0·577	0·579	0·216	0	0·444	0	0·567	0·940	0	0·769	0·873	0·386
−0·4	0·651	0·662	0	0·615	0·616	0·201	0	0·719	0	0·692	1·25	0	0·875	1·06	0·443
−0·45	0·733	0·748	0	0·661	0·661	0·186	0	1·15	0	0·830	1·65	0	0·983	1·29	0·500
−0·5	0·814	0·833	0	0·716	0·714	0·171	0	1·84	0	0·981	2·16	0	1·09	1·60	0·555

Finally we tested in which way the f_{ij} were influenced by a change of the scaling factors dx and dy. The results for the element defined by (33f) with $i = 5$ are given in Table VII. Obviously, all error constants are $0(\Delta^2)$.

Table VII. The dependence of f_{ij} on the gridsize for element (33f) with $i = 5$

	$-^{10}\log f_{12}$			$-^{10}\log f_{13}$			$-^{10}\log f_{21}$			$-^{10}\log f_{22}$			$-^{10}\log f_{23}$			$-^{10}\log f_{11}$
$dx = dy$	w_2	w_3	w_4	w_2	w_3	w_4	w_2	w_3	w_4	w_2	w_3	w_4	w_2	w_3	w_4	w_3
1	0·4	0·4	—	0·3	0·3	0·6	—	0·8	—	0·5	0·3	—	0·3	0·2	0·6	1·0
$10^{-1/2}$	1·4	1·4	—	1·3	1·3	1·6	—	1·9	—	1·5	1·3	—	1·2	1·2	1·6	2·0
10^{1}	2·4	2·4	—	2·3	2·3	2·5	—	2·9	—	2·5	2·3	—	2·2	2·2	2·5	3·0
$10^{1/2}$	3·4	3·4	—	3·3	3·3	3·5	—	3·9	—	3·5	3·3	—	3·2	3·2	3·5	4·0
10^{2}	4·4	4·4	—	4·3	4·3	4·5	—	4·9	—	4·5	4·3	—	4·2	4·2	4·5	5·0

The error in the approximation of known functions

The results of the previous subsection indicate that we may expect method W_4 to give the most accurate results in the approximation of the derivatives; using this method, the third derivatives of the function to be approximated occur in the truncation error with minimal constants. Here, we will investigate the actual error in the approximation, and for that end we have chosen the following set of test functions:

$$f_{i,j,\alpha}(x, y) = g_i(\alpha h_j(x, y)), \qquad i = 1, \ldots, 4, j = 1, 2, \alpha \in [10^{-3}, 1] \tag{35}$$

with

$$h_1(x, y) = x + y, h_2(x, y) = x - y/2, g_1 = \sin, g_2 = \cos, g_3 = \exp, g_4(x) = (1 + x)^3$$

For each test function f, a given element and a given approximation method, we computed a mixed error $\varepsilon(f)$ defined by

$$\varepsilon(f) = \tfrac{1}{5}\sqrt{\{(f_x - \mathbf{f}_x)^2 + (f_y - \mathbf{f}_y)^2 + \tfrac{1}{2}(f_{xx} - \mathbf{f}_{xx})^2 + \tfrac{1}{2}(f_{yy} - \mathbf{f}_{yy})^2 + (f_{xy} - \mathbf{f}_{xy})^2\}} \tag{36}$$

where $\mathbf{f}$ denotes the calculated approximation. To eliminate the influence of the function considered, we formed a mean value over the set of test functions given by

$$sd(\alpha) = \tfrac{1}{8} \sum_{i=1}^{4} \sum_{j=1}^{2} -\log(\varepsilon(f_{i,j,\alpha})) \tag{37}$$

Table VIII. The values of $sd(\alpha)$ for the element given by Frey, and a square element

	Element of Table IX			Square element
α	w_2	w_3	w_4	$w_2 = w_3 = w_4$
1	0·97	0·95	1·03	0·82
$10^{-1/2}$	2·32	2·09	2·72	2·47
10^{-1}	3·49	2·81	4·37	4·10
$10^{-3/2}$	4·63	3·46	6·00	5·73
10^{-2}	5·57	4·09	7·63	7·35
$10^{-5/2}$	6·88	4·72	9·25	8·98
10^{-3}	8·01	5·34	10·86	10·60

The values of $sd(\alpha)$ are listed in Table VIII; the approximations were made on an element given by Frey (see Table IX) and on the square element defined by (31) and (32). In the latter case the three methods gave identical results, as could be expected.

Table IX. The co-ordinates of the element given by Frey

i	x_i	y_i
1	−0·7500	0·2813
2	−0·1875	0·6563
3	0·5313	1·0000
4	−0.6250	−0·2813
5	0	0
6	0·7138	0·1875
7	−0·5625	−0·8433
8	0·0938	−0·7188
9	0·8125	−0·6875

Application to an elliptic problem

We considered the equation

$$\begin{aligned} \Delta u &= 2\,e^{x+y} \quad \text{on } \Omega \\ u(x, y) &= e^{x+y} \quad \text{on } \partial\Omega \end{aligned} \tag{38}$$

with the domain Ω given by (Figure 6): $\{(x, y)|\,|x| \le 1, -1 \le y \le 0\} \cup \{(x, y)|x^2 + y^2 \le 1\}$. The analytic solution is given by $u(x, y) = e^{x+y}$. We regarded the three straight lines and the arc as the four boundaries, and divided each one in N equal parts. Then we connected the corresponding points on the upper and lower boundary and defined the interior nodes to be

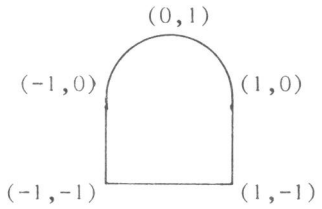


Figure 6. The domain Ω

$$z_{i,j} = \left(\frac{j}{N}\,\frac{2i-N}{N} - \frac{N-j}{N}\cos\frac{i\pi}{N},\ -\frac{j}{N} + \frac{N-j}{N}\sin\frac{i\pi}{N}\right)^{\mathrm{T}} \tag{39}$$

Of course we could have used a more sophisticated mesh generation scheme, but the subdivision given above is quite suitable for our testing purposes. Since the boundary conditions determine the values of u at the boundary nodes, we can set up a system of $(N-1)^2$ linear algebraic equations by using the discretization methods from the previous sections. The linear systems were solved by successive over-relaxation. In Table X we tabulated the differences between the analytic solution and the solution of the discrete problem, using the maximum norm. We note that the equations obtained were not all diagonally dominant. At least in the neighbourhood of

Table X. The maximum error in the approximation of the solution of equation (38)

	^{10}log max. error		
N	W_2	W_3	W_4
2	1·5	1·5	1·4
3	1·6	0·7	1·6
4	1·3	0·7	1·7
5	1·3	0·8	1·9
6	1·4	0·9	2·0
8	1·6	1·0	2·3
10	1·8	1·2	2·5
12	1·9	1·3	2·6
16	2·1	1·4	2·9
20	2·3	1·5	3·1

the left- and right-hand side boundaries, where the nine-point elements are very ill-shaped, off-diagonal elements of different sign occurred. Inside the region, where the elements resemble more or less rectangles, method W_4 turned out to produce much more diagonally dominant equations than the other two methods (115 vs. 75 in the case $N = 16$).

Finally, we solved the same problem on domains with sizes $1/\sqrt{10}$ and $1/10$ of the original ones. We list the results in Table XI, together with the computation time needed to set up the linear system, as measured on the Cyber 72 computer. Note, however, that these timings serve merely as an indication of the complexity of the methods; they were implemented only for testing purposes, and were not optimized with respect to efficiency.

Table XI. The maximum error and timing on domains with size $1/\sqrt{10}$ and $1/10$

	Size $1/\sqrt{10}$ ^{10}log max. error			Size $1/10$ ^{10}log max. error			Time (sec)		
N	W_2	W_3	W_4	W_2	W_3	W_4	W_2	W_3	W_4
2	2·4	2·4	3·4	3·3	3·5	5·4	0·02	0·02	0·11
4	2·5	1·7	3·6	3·6	2·4	5·1	0·17	0·05	1·00
8	2·9	2·0	4·1	4·0	2·7	5·7	0·91	0·27	5·66
16	3·4	2·3	4·7	4·5	2·9	6·5	4·24	1·18	25·9

Mixed boundary conditions

As we noted in remark 5, the least squares method can be applied to mixed boundary conditions, too. In order to test this implementation, we again considered equation (38). However, we replaced the boundary condition on the line $y = -1$ by

$$c \cdot u(x, y) + (1 - c)u_y(x, y) = e^{x+y}, \qquad y = -1 \tag{40}$$

Note, that we obtain conditions of Dirichlet and Van Neumann type, for $c = 1$ and $c = 0$, respectively. In Table XII we listed the results for some different values of the constants c and N (the number of gridlines).

Table XII. The maximum error in the approximation of equation (38) together with (40)

	^{10}log max. error					
N	$c = 1$	$c = 0{\cdot}8$	$c = 0{\cdot}6$	$c = 0{\cdot}4$	$c = 0{\cdot}2$	$c = 0$
2	1·4	1·5	1·6	1·7	1·9	2·7
4	1·7	1·8	1·8	1·8	1·8	1·8
8	2·3	2·3	2·3	2·3	2·2	2·3
16	2·9	2·9	2·9	2·9	2·9	2·9

REFERENCES

1. W. H. Frey, 'Flexible finite-difference stencils from isoparametric finite elements', *Int. J. num. Meth. Engng*, **11**, 1653–1665 (1977).
2. A. R. Gourlay, 'Splitting methods for time-dependent partial differential equations', in *Proc. 1976 Conf.: The State of Art in Numerical Analysis* (Ed. D. A. H. Jacobs), Academic Press (to appear).
3. P. J. Van der Houwen and J. Verwer, 'Non-linear splitting methods for semi-discretized parabolic differential equations', *Report NW 51/77*, Mathematisch Centrum, Amsterdam (1977).
4. J. Kok, P. J. Van der Houwen and P. H. M. Wolkenfelt, 'A semi-discretization algorithm for two-dimensional partial differential equations', *Report NW*, Mathematisch Centrum, Amsterdam (to appear).
5. *NAG Library Manual*, Mark 5, Oxford (1977).
6. N. L. Schryer, 'Numerical solution of time-varying partial differential equations in one space variable', *Comp. Science Techn. Rep. No. 53*, Bell Laboratories, Murray Hill (1975).
7. R. F. Sincovec and N. K. Madsen, 'Software for nonlinear partial differential equations', *ACM Trans. Mathematical Software*, **1**, 232–260 (1975).
8. J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.
9. J. H. Wilkinson and C. Reinsch, *Handbook for Automatic Computation*, vol. 2, Linear Algebra, Springer-Verlag, 1971.
10. N. N. Yanenko, *The Method of Fractional Steps*, Springer-Verlag, Berlin, 1971.

Generalized Runge-Kutta methods for coupled systems of hyperbolic differential equations

K. Dekker (*)

ABSTRACT

Runge-Kutta formulas are discussed for the integration of systems of differential equations. The parameters of these formulas are square matrices with component-dependent values. The systems considered are supposed to originate from hyperbolic partial differential equations, which are coupled in a special way. In this paper the discussion is concentrated on methods for a class of two coupled systems. For these systems first and second order formulas are presented, whose parameters are diagonal matrices. These formulas are further characterized by their low storage requirements, by a reduction of the computational effort per timestep, and by their relatively large stability interval along the imaginary axis. The new methods are compared with stabilized Runge-Kutta methods having scalar-valued parameters. It turns out that a gain factor of 2 can be obtained.

1. INTRODUCTION

Runge-Kutta methods for second order differential equations with prescribed initial values are well known in literature (e.g. ZONNEVELD [8], FEHLBERG [1]). When the first derivative does not occur in the second equation, these special methods are more efficient than comparable methods for first order equations; for example, they may attain a higher *order of accuracy* with the same amount of derivative evaluations, or may possess a larger *stability region* (VAN DER HOUWEN [5]). When the second order equation is transformed into a system of two first order equations, these special methods may be considered as *generalized Runge-Kutta methods* whose parameters are square matrices. Evidently, these generalized methods derive their usefulness from taking into account the special structure of the Jacobian matrix of the resulting equations.

In this paper we shall start to investigate generalized Runge-Kutta methods, which are not restricted to systems resulting from second order equations, but which apply to systems of the type

$$\frac{d\vec{y}_i}{dx} = \vec{f}_i(\vec{y}_1, \ldots, \vec{y}_k), \qquad i = 1, \ldots, k, \tag{1.1}$$

$\vec{y}_i$, $i = 1, \ldots, k$, being prescribed at $x = x_0$. We observe that each component of this system in itself is a vector of a certain length, which is not necessarily the same for each component. Systems of this type may arise by applying the method of lines to a coupled system of hyperbolic or parabolic partial differential equations. When the Jacobian matrix of (1.1), given by

$$J_{ij} = \frac{\partial \vec{f}_i}{\partial \vec{y}_j}, \qquad i = 1, \ldots, k, \quad j = 1, \ldots, k, \tag{1.2}$$

is sparse, it is likely that generalized Runge-Kutta methods are more efficient than ordinary Runge-Kutta methods.

In the next sections we shall describe the generalized Runge-Kutta method, and derive conditions for consistency (up to order 2) and for low storage requirements. The stability analysis is performed by imposing conditions on the Jacobian matrix, which are fulfilled for a wide class of hyperbolic systems. This particular choice is motivated by the fact that we want to investigate generalized Runge-Kutta methods for the *two-dimensional shallow water equation* (KREIS [6]) in a forthcoming paper.

In section 4 we restrict ourselves to problems consisting of two coupled systems ($k = 2$ in 1.1), of which $\vec{f}_2$ does not depend on $\vec{y}_2$. Second order, m-point formulas using two or three arrays of storage are constructed. In the latter case the resulting stability condition reads

$$h_n \leq \frac{m-1}{\sigma(J)}, \quad m \text{ odd}. \tag{1.3}$$

Here $\sigma(J)$ denotes the spectral radius of the Jacobian matrix J. The number of derivative evaluations per time step for these formulas, however, is less than m,

(*) K. Dekker, Stichting Mathematisch Centrum, 2e Boerhaavestraat 49, Amsterdam 1005, The Netherlands

viz. $\frac{m+2}{2}$, so that we effectively have a stability limit of $2(m-1)/(m+2)$. Thus, asymptotically a factor 2 is gained over ordinary stabilized second order Runge-Kutta methods, which have an effective stability limit of $(m-1)/m$ (VAN DER HOUWEN [3]). In the near future numerical results will be reported obtained by the new formulas, applied to the wave equation and the equation of the flow in a narrow canal. Also, we intend to construct generalized methods for problems consisting of three coupled systems.

2. THE GENERALIZED RUNGE-KUTTA METHOD

Consider the system of differential equations (1.1). In order to simplify the notation we introduce the variables

$$y = (\vec{y}_1, \ldots, \vec{y}_k)^T \text{ and } f(y) = [\vec{f}_1(y), \ldots \vec{f}_k(y)]^T. \tag{2.1}$$

The generalized m-point Runge-Kutta method is defined by

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

$$y_{n+1}^{(j)} = M_j y_n + h_n \sum_{l=0}^{j-1} N_{jl} f(y_{n+1}^{(l)}), \quad j = 1, \ldots, m,$$

$$y_{n+1} = y_{n+1}^{(m)}. \tag{2.2}$$

Here, y_{n+1} denotes the numerical approximation to the solution y at the point $x = x_n + h_n$. The quantities M_j and N_{jl} stand for $k \times k$ matrices, whose entries are matrices, too, the size depending on the dimensions of $\vec{y}_1, \ldots, \vec{y}_k$.

Example 2.1

Consider the method for second order differential equations

$$\frac{d^2\vec{w}}{dx^2} = \vec{g}(\vec{w}, \frac{d\vec{w}}{dx}), \tag{2.3}$$

described by ZONNEVELD [1964]:

$$\vec{z}_n = \vec{w}_n',$$

$$\vec{w}_{n+1}^{(1)} = \vec{w}_n + h_n\vec{z}_n, \quad \vec{z}_{n+1}^{(1)} = \vec{z}_n + h_n\vec{g}(\vec{w}_n, \vec{z}_n),$$

$$\vec{w}_{n+1} = \vec{w}_n + h_n\vec{z}_n + \frac{1}{2}h_n^2\vec{g}(\vec{w}_n, \vec{z}_n),$$

$$\vec{w}_{n+1}' = \vec{z}_n + \frac{1}{2}h_n\{\vec{g}(\vec{w}_n, \vec{z}_n) + \vec{g}(\vec{w}_{n+1}^{(1)}, \vec{z}_{n+1}^{(1)})\}.$$

When we define $y = (\vec{w}, \vec{z})^T$ and $f = (\vec{z}, \vec{g})^T$, this method can be represented by scheme (2.2) where m = 2 and

$$M_1 = M_2 = \begin{vmatrix} I & h_nI \\ 0 & I \end{vmatrix}, \quad N_{10} = \begin{vmatrix} 0 & 0 \\ 0 & I \end{vmatrix}$$

$$N_{20} = \begin{vmatrix} 0 & \frac{1}{2}h_nI \\ 0 & \frac{1}{2}I \end{vmatrix}, \quad N_{21} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{1}{2}I \end{vmatrix} \tag{2.4a}$$

In these expressions I denotes the unity matrix of appropriate order. The occurrence of this matrix in an off-diagonal position is allowed, because the vectors $\vec{w}$ and $\vec{z}\,(= \vec{w}')$ have the same number of components. Note, however, that the representation (2.4a) is not unique. Another choice for the parameter matrices reads

$$M_1 = M_2 = N_{10} = \begin{vmatrix} 1 & 0 \\ 0 & I \end{vmatrix}, \quad N_{20} = N_{21} = \begin{vmatrix} \frac{1}{2}I & 0 \\ 0 & \frac{1}{2}I \end{vmatrix}, \tag{2.4b}$$

so that the method reduces to an ordinary Runge-Kutta method for a system of equations.
A less trivial example is given in VAN DER HOUWEN [5].

Example 2.2

Consider the method for second order differential equations without first derivatives, described by

$$\vec{w}_{n+1} = \vec{w}_n + h_n\vec{w}_n' + \frac{1}{2}h_n^2\vec{g}[\vec{w}_n + \frac{1}{2}h_n\vec{w}_n' + \frac{1}{16}h_n^2\vec{g}(\vec{w}_n + \frac{1}{2}h_n\vec{w}_n')],$$

$$\vec{w}_{n+1}' = \vec{w}_n' + h_n\vec{g}[\vec{w}_n + \frac{1}{2}h_n\vec{w}_n' + \frac{1}{16}h_n^2\vec{g}(\vec{w}_n + \frac{1}{2}h_n\vec{w}_n')]. \tag{2.5}$$

Using the same conventions as in the previous example, we can represent this method by m = 3 and

$$M_1 = M_2 = \begin{vmatrix} I & \frac{1}{2}h_nI \\ 0 & I \end{vmatrix}, \quad M_3 = \begin{vmatrix} I & h_nI \\ 0 & I \end{vmatrix},$$

$$N_{21} = \begin{vmatrix} 0 & \frac{1}{16}h_nI \\ 0 & 0 \end{vmatrix}, \quad N_{32} = \begin{vmatrix} 0 & \frac{1}{2}h_nI \\ 0 & I \end{vmatrix},$$

$$N_{10} = N_{20} = N_{30} = N_{31} = 0, \tag{2.5a}$$

or, alternatively by m = 5 and

$$M_1 = M_2 = M_3 = M_4 = M_5 = \begin{vmatrix} I & 0 \\ 0 & I \end{vmatrix},$$

$$N_{10} = N_{32} = \begin{vmatrix} \frac{1}{2}I & 0 \\ 0 & 0 \end{vmatrix}, \quad N_{21} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{1}{8}I \end{vmatrix},$$

$$N_{43} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{1}{2}I \end{vmatrix}, \quad N_{53} = \begin{vmatrix} 0 & 0 \\ 0 & I \end{vmatrix},$$

$$N_{54} = \begin{vmatrix} I & 0 \\ 0 & 0 \end{vmatrix}, \quad N_{20} = N_{30} = N_{31} = N_{40} = N_{41} = N_{42} = N_{50} = N_{51} = N_{52} = 0. \tag{2.5b}$$

We remark that, although (2.5b) defines a five-point formula, it evidently requires only two evaluations of the second derivative g, namely those with $y_{n+1}^{(1)}$ and $y_{n+1}^{(3)}$ as arguments.

3. CONDITIONS FOR THE PARAMETER MATRICES

In this section we shall derive the conditions which should be imposed on the matrices M_j and N_{jl} in order to ensure second order consistency, minimal storage requirements and stability. Whereas we intend to derive schemes which are applicable to a restricted class of equations of type (1.1), we shall formulate the conditions in terms of the variables y_n, $f_n [= f(y_n)]$ and J_n, the Jacobian matrix at (x_n, y_n). In general, the conditions are very complicated; therefore we shall simplify them by inposing the following restrictions on the parameter matrices :

(3.1) The matrices M_j and N_{jl}, $j = 1, \ldots, m$, $l = 0, \ldots, j-1$, do not depend on the Jacobian J_n.

(3.2) The matrices M_j satisfy the relation $M_j = I + 0(h_n)$, and the matrices N_{jl} satisfy $h_n \times N_{jl} = 0(h_n)$, $j = 1, \ldots, m$, $l = 0, \ldots, j-1$.

The examples of section 2 show that (3.2) need not be a severe restriction, whereas generalized RK-schemes whose parameter matrices depend on the Jacobian have already been analysed by several authors.

3.1. Consistency conditions

The order equations for scheme (2.2) can be derived in the usual way (see e.g. ZONNEVELD [8]) by expanding y_{n+1} and the analytic solution of (1.1) through the point $y(x_n) = y_n$ in a Taylor-series in h_n, and equating the corresponding terms.
The conditions for orders p up to 2 are listed in table 3.1. It should be remarked that table 3.1 presents for p = 2 "additional" conditions, i.e. the condition for second order consistency are the conditions listed for both p = 1 and p = 2.

The conditions given in table 3.1 determine the consistency of scheme (2.2) for a particular differential equation at a specific point. Requiring that (2.2) is consistent for all problems of type (1.1) yields the conditions listed in table 3.2; these conditions can easily be derived from table 3.1 by suitable substitutions for y_n, f_n and J_n.

Table 3.2. Consistency conditions for scheme (2.2)

p = 1	$M_m^{(1)} = 0,$ $\sum_{l=0}^{m-1} N_{ml}^{(0)} = I.$
p = 2	$M_m^{(2)} = 0,$ $\sum_{l=0}^{m-1} N_{ml}^{(1)} = 0,$ $\sum_{l=0}^{m-1} N_{ml}^{(0)}(p,q)\, M_l^{(1)}(r,s) = 0,$ $p, q, r, s \in [1, \ldots, k]$ $\sum_{l=0}^{m-1} N_{ml}^{(0)}(p,q) \sum_{n=0}^{l-1} N_{ln}^{(0)}(r,s) = \frac{1}{2}\delta_{pq}\delta_{rs},$ $p, q, r, s \in [1, \ldots, k]$ Here, $N_{ml}(p,q)$ denotes the element in row p and column q of the matrix N_{ml}, whereas δ stand for the Kronecker function.

Example 3.1.

The scheme determined by (2.5a) does not satisfy the conditions for p = 1 given in table 3.2. As a consequence, scheme (2.5a) is generally not consistent of order one,

Table 3.1. Consistency conditions for scheme (2.2), applied to equation (1.1)

p = 1	$M_m^{(1)} y_n + \sum_{l=0}^{m-1} N_{ml}^{(0)} f_n = f_n$
p = 2	$\frac{1}{2} M_m^{(2)} y_n + \sum_{l=0}^{m-1} N_{ml}^{(1)} f_n + \sum_{l=0}^{m-1} N_{ml}^{(0)} J_n M_l^{(1)} y_n + \sum_{l=0}^{m-1} N_{ml}^{(0)} J_n \sum_{r=0}^{l-1} N_{lr}^{(0)} f_n = \frac{1}{2} J_n f_n,$ where $M_l^{(i)} = \{\frac{d^i}{dh^i} M_l(h_n)\}_{h_n = 0}$, $i = 0, 1, 2$, $l = 1, \ldots, m,$ and $N_{ml}^{(i)} = \{\frac{d^i}{dh^i} N_{ml}(h_n)\}_{h_n = 0}$, $i = 0, 1$, $l = 0, \ldots, m-1.$

when it is applied to an arbitrary system. However, scheme (2.5a) satisfies the conditions given in table 3.1, if the following equalities hold :

$$\begin{vmatrix} 0 & I \\ 0 & 0 \end{vmatrix} y_n = \begin{vmatrix} I & 0 \\ 0 & 0 \end{vmatrix} f_n \quad \text{and}$$

$$\begin{vmatrix} 0 & I \\ 0 & 0 \end{vmatrix} f_n + \begin{vmatrix} 0 & 0 \\ 0 & I \end{vmatrix} J_n \begin{vmatrix} I & 0 \\ 0 & 0 \end{vmatrix} f_n = J_n f_n .$$

These equations are evidently satisfied when we substitute $y_n = (\vec{w}_n, \vec{w}_n')$, $f_n = [\vec{w}_n', \vec{g}(\vec{w}_n)]$ and

$J_n = \begin{pmatrix} 0 & I \\ \frac{dg}{dwn} & 0 \end{pmatrix}$, so (2.5a) is consistent of order 2 for second-order differential equations without first derivatives.

In a similar way one easily verifies that scheme (2.5b) is only consistent if order two is the equality

$$\begin{vmatrix} 0 & 0 \\ 0 & I \end{vmatrix} J_n \begin{vmatrix} I & 0 \\ 0 & 0 \end{vmatrix} + \begin{vmatrix} I & 0 \\ 0 & 0 \end{vmatrix} J_n \begin{vmatrix} 0 & 0 \\ 0 & I \end{vmatrix} = J_n$$

holds, which again is the case when second order equations without first derivatives are considered.

3.2. Storage requirements

As we intend to apply the schemes to large systems originating from partial differential equations, attention should be paid to the storage requirements. We shall derive here the conditions for schemes requiring two and three arrays of storage (see VAN DER HOUWEN [2]).

Schemes requiring two arrays of storage

The flow of computation in schemes which require only two arrays of storage might be represented by the flow chart

$$\begin{array}{ccc} y_{n+1}^{(j-1)} & \rightarrow & y_{n+1}^{(j)} \\ \downarrow & \nearrow & \\ f(y_{n+1}^{(j-1)}) & & \end{array}$$

or by the formula

$$y_{n+1}^{(j)} = D_j \{ y_{n+1}^{(j-1)} + h_n E_j f(y_{n+1}^{(j-1)}) \}, \qquad j = 1, \ldots, m, \tag{3.3}$$

where D_j denotes a general matrix and E_j a sufficiently simply matrix in order to compute the product without auxiliary storage.

Comparing (3.3) with scheme (2.2), we obtain the following relations for M_j and N_{jl} :

$$M_j = \prod_{l=1}^{j} D_l, \qquad j = 1, \ldots, m,$$

$$N_{jl} = \prod_{r=l+1}^{j} D_r E_{l+1}, \qquad \begin{matrix} j = 1, \ldots, m, \\ l = 0, \ldots, j-1, \end{matrix} \tag{3.4}$$

Elimination of D_j and E_j yields

$$N_{jl} = M_j M_{l+1}^{-1} N_{l+1,l} \qquad \begin{matrix} j = 1, \ldots, m, \\ l = 0, \ldots, j-1. \end{matrix} \tag{3.5}$$

We note that M_{l+1}^{-1} exists for sufficiently small h_n, in view of relation (3.2).

Schemes requiring three arrays of storage

Introducing an additional set of vectors z_{n+1}, we can construct schemes of type (2.2) by means of recurrence relations of the type

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

$$y_{n+1}^{(j)} = A_j y_{n+1}^{(j-1)} + h_n E_j f(y_{n+1}^{(j-1)}) + B_j z_{n+1}^{(j-1)},$$

$$z_{n+1}^{(j)} = C_j y_{n+1}^{(j-1)} + h_n F_j f(y_{n+1}^{(j-1)}) + D_j z_{n+1}^{(j-1)},$$

$$j = 1, \ldots, m,$$

$$y_{n+1} = y_{n+1}^{(m)} \tag{3.6}$$

Assuming A_j non-singular (this is implied by condition (3.2)), we may set $C_j = 0$ without loss of generality. In fact, given a recurrence relation (3.6), it is easy to construct an equivalent relation (yielding the same y_{n+1}) with $C_j = 0$. Comparing (3.6) with scheme (2.2), we obtain the following relations for M_j and N_{j1} :

$$M_j = \prod_{l=1}^{j} A_l,$$

$$N_{jl} = M_j M_{l+1}^{-1} N_{l+1,l} + \sum_{r=l+2}^{j} M_j M_r^{-1} B_r \prod_{s=l+2}^{r-1} D_s F_{l+1},$$

$$N_{jj-1} = E_j, \qquad \begin{matrix} l = 0, \ldots j-2 \\ j = 1, \ldots, m. \end{matrix} \tag{3.7}$$

We remark that, in general, the matrices D_j cannot be eliminated from this formula, as they might be singular. However, it is easily verified that for a suitable transformation of $z_{n+1}^{(j)}$ the matrices D_j will be of the form

$$\begin{vmatrix} I & X \\ 0 & 0 \end{vmatrix}$$

3.3. Stability requirements

To analyse the stability of scheme (2.2), we study the effect of a perturbation Δy_n of y_n on the resulting

vector y_{n+1}. Let J_n denote the Jacobian matrix of the right hand side $f(y_n)$; then this perturbation is approximately given by

$$\Delta y_{n+1} = M_j \Delta y_n + \sum_{l=0}^{j-1} h_n N_{jl} J_n \Delta y_n^{(l)}, \quad j = 1, \ldots, m, \tag{3.8}$$

or alternatively by

$$\Delta y_{n+1}^{(j)} = R_j(h_n J_n) \Delta y_n, \quad j = 1, \ldots, m,$$

$$R_j(h_n J_n) = M_j + \sum_{l=0}^{j-1} h_n N_{jl} J_n R_l(h_n J_n). \tag{3.8a}$$

We shall call method (2.2) stable if all the eigenvalues of $R_m(h_n J_n)$ are within the unit circle; when one or more eigenvalues are on the unit circle, the method will be called weakly stable. Integrating problems with a constant Jacobian with a stable method, the effect of a perturbation Δy_n will ultimately be damped out, as the k-th power of the amplification matrix $R_m(h_n J)$ will tend to the zero matrix as k tends to infinity. Using a weakly stable method, the effect of Δy_n will grow less than exponentially, the rate of growth depending on the number of coinciding eigenvalues on the unit circle.
Next, we consider a finite interval of integration and let h_n tend to zero. Then, the eigenvalues of the matrix $R_m(h_n J)$ with multiplicity greater than one may tend to one, as is illustrated in the following example.

Example 3.2

Consider the second order method generated by (m=2):

$$M_1 = M_2 = \begin{vmatrix} 1 & h \\ 0 & 1 \end{vmatrix}, \quad N_{10} = N_{21} = \begin{vmatrix} 0 & 0 \\ 0 & 1 \end{vmatrix}, \quad N_{20} = \begin{vmatrix} 0 & \frac{1}{2}h \\ 0 & \frac{1}{2} \end{vmatrix}. \tag{3.9}$$

Application of this method to the differential equations

$$\frac{dy}{dx} = z, \quad \frac{dz}{dx} = -4y - 4z, \quad y(0) = y_0, \quad z(0) = z_0, \tag{3.10}$$

yields the recurrence relation

$$\begin{vmatrix} y_{n+1} \\ z_{n+1} \end{vmatrix} = \frac{1}{5} \begin{vmatrix} 1 & 2 \\ -2 & 1 \end{vmatrix} \begin{vmatrix} 1-2h+h^2 & 5h-10h^2 \\ 0 & 1-2h+h^2 \end{vmatrix} \begin{vmatrix} 1 & -2 \\ 2 & 1 \end{vmatrix} \begin{vmatrix} y_n \\ z_n \end{vmatrix}$$

$$= \tilde{S}(h) \begin{vmatrix} y_n \\ z_n \end{vmatrix}. \tag{3.11}$$

Although $\tilde{S}(h)$ has a multiple eigenvalue, $\|\tilde{S}(h)^n\|$ is bounded by the constant $1 + 5hn$, as the off-diagonal elements of its Jordan-normal form are of order h. This suggests that we should consider amplification matrices, whose Jordan form have off-diagonal elements of order h. In the following lemma, we will show that scheme (2.2) has this property, provided that (3.1) and (3.2) are satisfied.

LEMMA 3.1
The amplification matrix belonging to a generalized Runge-Kutta method which satisfies the conditions (3.1) *and* (3.2), *has a Jordan normal form with elements of order h in off-diagonal position.*

Proof

From the definition of the amplification matrix $R_m(h_n J_n)$ in (3.8a) and the conditions (3.1) and (3.2) it is obvious that there exists a matrix A, such that

$$R_m(h_n J_n) = I + A \text{ and } \|A\|_2 = 0(h_n).$$

Now, let B be the Jordan normal form of A, given by the unitary transformation $B = T A T^{-1}$. From $|B_{ij}| \leq \|Be_j\|_2 \leq \|B\|_2 = \|A\|_2 = 0(h_n)$ follows that all elements of B are of order h. As the Jordan normal form of $R_m(h_n J_n)$ is given by I + B, it is clear that all the off-diagonal elements of this Jordan form are of order h.

COROLLARY
The global discretization error of a generalized Runge-Kutta method for $h \to 0$ *increases at most linearly with the number of steps, if the conditions* (3.1) *and* (3.2) *are satisfied, and the amplification matrix has eigenvalues on or within the unit circle.*

The above corollary suggests us to verify the stability of a generalized Runge-Kutta method for a given problem by proving that the eigenvalues of the amplification matrix are in modulus less or equal to one. However, this task is not as simple as in the case of ordinary Runge-Kutta methods, as was already observed by VAN DER HOUWEN [5]. The reason for this is that we cannot reduce a system of equations to a set of single equations, which are more easily analyzed, because the eigenvectors of the Jacobian matrix differ in general from the eigenvectors of the amplification matrix. This behaviour may be illustrated in the following example :

Example 3.3

Consider the generalized Runge-Kutta method defined by m = 2 and

$$M_1 = M_2 = I, \quad N_{10} = \begin{vmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{vmatrix}, \quad N_{20} = 0, \quad N_{21} = I. \tag{3.12}$$

Application of this method to the *model problem for hyperbolic equations*

$$\frac{dy}{dx} = -cz, \quad \frac{dz}{dx} = cy, \quad y(0) = y_0, \quad z(0) = z_0, \tag{3.13}$$

yields the recurrence relation

$$\begin{vmatrix} y_{n+1} \\ z_{n+1} \end{vmatrix} = \begin{vmatrix} 1 - \frac{1}{2}h^2c^2 - hc \\ hc \quad 1 - h^2c^2 \end{vmatrix} \begin{vmatrix} y_n \\ z_n \end{vmatrix}, \quad n = 0, \ldots. \tag{3.14}$$

The eigenvalues of the amplification matrix are

$$\lambda_{1,2} = 1 - \frac{3}{4} h^2 c^2 \pm i hc \sqrt{1 - \frac{1}{4} h^2 c^2} \; ;$$

these eigenvalues are in modulus less than or equal to 1 if $0 \leqslant hc \leqslant 1$.
When we try to uncouple system (3.13) by introducing the eigenvectors of the Jacobian matrix, $u = (1-i, 1-i)^T$, $v = (1+i, 1-i)^T$, we can rewrite (3.13) and (3.14) as

$$\frac{du}{dx} = -icu, \quad \frac{dv}{dx} = +icv, \quad u(0) = u_0 = (1+i)y_0 + (1-i)z_0,$$

$$v(0) = v_0 = (1-i)y_0 + (1+i)z_0, \tag{3.13a}$$

and

$$\begin{vmatrix} u_{n+1} \\ v_{n+1} \end{vmatrix} = \begin{vmatrix} 1 - \frac{3}{4}h^2c^2 + ihc - \frac{1}{4}ih^2c^2 & \\ \frac{1}{4}ih^2c^2 & 1 - h^2c^2 - ihc \end{vmatrix} \begin{vmatrix} u_n \\ v_n \end{vmatrix},$$

$$n = 0, \ldots . \tag{3.14a}$$

Evidently, the amplification matrix of (3.14a) is not a simple diagonal matrix, with as elements polynomials in ihc, as one would obtain in the case of ordinary Runge-Kutta methods.
From this example we may conclude that the stability analysis of a generalized scheme in terms of the eigenvalues of the Jacobian matrix is in general impossible. In order to derive a priori stability properties of a generalized scheme, we shall restrict ourselves to a class of differential equations which is characterized by the fact that the Jacobian matrix has pairs of purely imaginary eigenvalues. This particular choice is motivated by the observation that this situation frequently occurs after discretization of partial differential equations of hyperbolic type. For equations of this type the following lemma may be applied.

LEMMA 3.2
Let $R(hJ_n)$ *be the amplification matrix of a generalized scheme* (2.2) *applied to a system of equations with Jacobian matrix* J_n *of order* 2N. *Assume that the eigenvectors of* J_n *can be split into pairs* (u_k, v_k), *having eigenvalues* λ_k *and* $\bar{\lambda}_k$, *such that*

$$R(hJ_n)(u_k, v_k) = (u_k, v_k) \begin{vmatrix} P(h\lambda_k) & \overline{Q(h\lambda_k)} \\ Q(h\lambda_k) & P(h\lambda_k) \end{vmatrix}$$

$$= (u_k, v_k) A(h\lambda_k), \qquad k = 1, \ldots, N, \tag{3.15}$$

where (u_k, v_k) *denotes the matrix consisting of the columns of* u_k *and* v_k *and* P *and* Q *are polynomials. Then all the eigenvalues of* $R(hJ_n)$ *are in modulus less than or equal to* 1, *if both eigenvalues of all matrices* A_k *are in modulus less than or equal to* 1.

Proof

According to the assumption there exists a similarity transformation

$$TJ_nT^{-1} = \begin{vmatrix} \lambda & & & 0 \\ 1 & \bar{\lambda} & & \\ & 1 & \ddots & \\ & & \ddots & \lambda_N \\ 0 & & & \bar{\lambda}_N \end{vmatrix},$$

where the columns of T are formed by the eigenvectors u_k and v_k, $k = 1, \ldots, N$. Using (3.15) we find

$$R(hJ_n)T = T \begin{vmatrix} A_1 & & 0 \\ & A_2 & \\ & \ddots & \\ 0 & & A_N \end{vmatrix},$$

so that $T^{-1}R(hJ_n)T$ transforms $R(hJ_n)$ into a (2×2) block-diagonal matrix. Thus each eigenvalue of $R(hJ_n)$ corresponds to an eigenvalue of A_k, for some index k, which implies the assertion of the lemma.
Now we can define the *stability region* S of a scheme for which (3.15) holds as the region in the complex plane of z values, for which the eigenvalues of A(z) are within the unit circle. In particular we shall be interested in the *imaginary stability boundary* β_{im}; that is the largest positive number such that $0 \leqslant z \leqslant \beta_{im}$ implies $iz \in S$. The most simply way to find S and β_{im} is the application of the Hurwitz criterion : *the roots of the equation*

$$a^2 - Sa + P = 0$$

lie within or on the unit circle when the coefficients S *and* P *are real and satisfy the inequalities*

$$|S| \leqslant P + 1, \quad P \leqslant 1.$$

Application to (3.15) yields the *stability conditions*

$$|P(z)|^2 - |Q(z)|^2 \leqslant 1 \tag{3.16}$$

and

$$2 \operatorname{Re} P(z) \leqslant |P(z)|^2 - |Q(z)|^2 + 1.$$

Example 3.4

When we consider the method described in example 3.3, we find $P(z) = 1 + \frac{3}{4}z^2 + z$ and $Q(z) = -\frac{1}{4}iz^2$. Substitution into the inequalities (3.16) yield the conditions

$$\frac{1}{2}z^2 + \frac{1}{2}z \leqslant 0$$

and

$$z^2 \leqslant \frac{1}{2}z^4,$$

These conditions are satisfied for $z = ia$, $a \leqslant 1$, so we find $\beta_{im} = 1$.

In lemma 3.2 we did choose a special form for the matrices A in order to obtain as characteristic equation a polynomial with real coefficients, on which the Hurwitz criterion was applicable. We might have chosen for the elements of A four different polynomials in $h\lambda_k$, and then we might have applied the Schur criterion (see e.g. LAMBERT [7]) on the complex characteristic polynomial of $A(h\lambda_k)$, thus relaxing the conditions of the lemma. However, the formulation chosen is more simple, and seems to leave enough freedom in the choice of the parameter matrices.

We now consider the question under what conditions the amplification matrix $R(hJ_n)$ can be written in the form (3.15). Obviously, a sufficient condition is, that all matrices M_j and N_{jl}, $j = 1, \ldots, m$, $l = 0, \ldots, j-1$, are of the form

$$T \begin{vmatrix} aI & \overline{bI} \\ bI & \overline{aI} \end{vmatrix} T^{-1}, \qquad (3.17)$$

(T the matrix of eigenvectors of J_n, as in lemma 3.2). Substitution of these matrices in (3.8a) will yield (3.15). Thus the stability conditions (3.16) can be applied to generalized schemes of which the matrices are generated by (3.17), and the stability problem is reduced to the problem of finding suitable polynomials $P(z)$ and $Q(z)$.

In the following section we shall construct some pairs of polynomials, which are optimal in the sense that β_{im} is maximized. Here, we remark that the resulting scheme may be of little value if the matrices generated by (3.17) are not very sparse. However, for a model problem the matrices (3.17) may turn out to be sparse, and we may hope that the thus constructed scheme has good stability properties for less trivial problems, too.

Example 3.5

Assume that the Jacobian matrix J_n of a problem has a matrix of normalized eigenvectors T, which consists of 4 blocks,

$$T = \begin{vmatrix} U & U \\ V & -V \end{vmatrix}. \qquad (3.18)$$

Then it is easily verified that the matrices $T \begin{vmatrix} aI & bI \\ bI & aI \end{vmatrix} T^{-1}$ are sparse. In fact, we obtain

$$\begin{vmatrix} U & U \\ V & -V \end{vmatrix} \begin{vmatrix} aI & bI \\ bI & aI \end{vmatrix} \begin{vmatrix} U^H & V^H \\ U^H & -V^H \end{vmatrix} = \begin{vmatrix} \frac{1}{2}(a+b)I & 0 \\ 0 & \frac{1}{2}(a-b)I \end{vmatrix},$$

being a diagonal matrix; the number of non-zero elements may be further reduced by the choices $a = b$ and $a = -b$.

Matrices of the form $\begin{pmatrix} 0 & B \\ C & 0 \end{pmatrix}$, with purely imaginary eigenvalues, have property (3.18), for we may choose U to be the eigensystem of the matrix BC, and V as $\Lambda^{-1}CU$, where Λ is the matrix of eigenvalues of $\begin{pmatrix} 0 & B \\ C & 0 \end{pmatrix}$. The stability analysis of problems of this type may thus be performed by analyzing the stability of the model problem (3.13) with appropriate values of c (equal to the modulus of the eigenvalues of the Jacobian matrix).

4. GENERALIZED SECOND ORDER FORMULAS FOR A RESTRICTED CLASS OF EQUATIONS

In this section we shall construct m-point formulas, which are of second order for problems with a Jacobian matrix of the form $J = \begin{vmatrix} J_{11} & J_{12} \\ J_{21} & 0 \end{vmatrix}$. We only considered formulas using at most two or three arrays of storage; amongst those we tried to optimize the *effective imaginary stability boundary*, the quotient of the imaginary stability boundary and the number of derivative evaluations (which need not be equal to m, as was shown in example 2.2). The optimization was done by choosing optimal pairs of polynomials $P(z)$ and $Q(z)$. Schemes corresponding to these polynomials are found by (3.17), setting

$$T = \begin{vmatrix} (1+i)I & (1-i)I \\ (1-i)I & (1+i)I \end{vmatrix},$$

the matrix of eigenvectors of problem (3.13).

4.1. Stabilized second order formulas

We assume that the Jacobian matrix J may be written as a 2 by 2 matrix with matrices as entries (possibly originating from a system of two partial differential equations) and that J_{22} is the zero matrix. Considering generalized schemes (2.2) with $k = 2$, we derive from table 3.2 (and partially 3.1) the relations for second order consistency :

$$\begin{aligned} &M_m^{(1)} = 0, \quad M_m^{(2)} = 0 \\ &\sum_{l=0}^{m-1} N_{ml}^{(0)} = I, \quad \sum_{l=0}^{m-1} N_{ml}^{(1)} = 0 \\ &\sum_{l=0}^{m-1} N_{ml}^{(0)} \, (p, q) \, M_l^{(1)}(r, s) = 0, \\ &\qquad p, q, r, s \in \{1, 2\} \quad q + r \neq 4, \\ &\sum_{l=0}^{m-1} N_{ml}^{(0)} \, (p, q) \sum_{k=0}^{l-1} N_{lk}^{(0)}(r, s) = \frac{1}{2}\delta_{pq}\delta_{rs}, \\ &\qquad p, q, r, s \in \{1, 2\} \quad q + r \neq 4. \end{aligned} \qquad (4.1)$$

From definition (3.8a) it follows that the amplification matrix is given by

$$R_m(hJ_n) = M_m + h \sum_{l=0}^{m-1} N_{ml} J_n R_l(hJ_n)$$

$$= M_m + h \sum_{l=0}^{m-1} N_{ml} J_n M_l$$

$$+ h^2 \sum_{l=0}^{m-1} N_{ml} J_n \sum_{k=0}^{l-1} N_{lk} J_n M_k + h^3 \ldots,$$

so that, using (3.1), (3.2) and (4.1) we find

$$R_m(hJ_n) = I + h J_n + \tfrac{1}{2} h^2 J_n^2 + 0(h^3). \qquad (4.2)$$

Assuming that notation (3.15) is applicable, we see that the polynomials P and Q can be written as

$$P(z) = 1 + z + \tfrac{1}{2} z^2 + p_3 z^3 + \ldots + p_m z^m,$$

$$Q(z) = q_3 z^3 + \ldots + q_m z^m. \qquad (4.3)$$

Now, we try to choose the parameters $p_3, \ldots, p_m$ and $q_3, \ldots, q_m$ in such a way, that the conditions (3.16) are fulfilled for $0 \leqslant -iz \leqslant \beta_{im}$, for a value of β_{im} as large as possible.

2-point formulas

For a 2-point formula we have $P_2(z) = 1 + z + \frac{1}{2} z^2$ and $Q_2(z) = 0$, and substitution in (3.16) shows that the Hurwitz conditions are violated for small imaginary values of z.

3-point formulas

Substitution of $P_3(z)$ and $Q_3(z)$ in (3.16) yields the conditions

$$1 + z^4 (\tfrac{1}{4} - 2p_3) + z^6 (q_3^2 - p_3^2) \leqslant 1$$

and

$$|2 + z^2| \leqslant 2 + z^4 (\tfrac{1}{4} - 2p_3) + z^6 (q_3^2 - p_3^2).$$

It is easily verified that the choice $p_3 = q_3 = \frac{1}{8}$ results in an optimal stability boundary $\beta_{im} = 2$.

Multi-point formulas

Let us define the polynomials

$$V_m(z^2) = |P_m(z)|^2 - |Q_m(z)|^2 = 1 + v_4 z^4 + \ldots + v_{2m} z^{2m}$$

$$W_m(z^2) = 2 \operatorname{Re} P_m(z) = 2 + z^2 + w_4 z^4 + \ldots + w_{2k} z^{2k}, \quad 2k \leqslant m \qquad (4.4)$$

We now can express the Hurwitz-conditions (3.16) in terms of V_m and W_m as follows :

$$V_m(s) \leqslant 1 \qquad (s = z^2 \leqslant 0)$$

$$W_m(s) \leqslant V_m(s) + 1,$$

$$-W_m(s) \leqslant V_m(s) + 1. \qquad (4.5)$$

An optimum is achieved for

$$V_m(s) = 1 \text{ and } W_m(s) = 2\, T_k\left(\frac{2s^2}{\beta} + 1\right), \qquad (4.6)$$

where $\beta = 4k^2$, and $T_k(x)$ is the Chebyshev polynomial of degree k.

Unfortunately, we cannot find P_m and Q_m related according (4.4) to the polynomials V_m and W_m as given by (4.6) for all values of m. However, for odd values of m we did find polynomials satisfying (4.4), namely

$$P_m(z) = \frac{1}{z} \{1 + z + \tfrac{1}{2} z^2) T_k\left(\frac{z^2}{2k^2} + 1\right) - 1 - \tfrac{1}{8} z^4 \}$$

$$Q_m(z) = P_m(z) - 1 - z - \tfrac{1}{2} z^2. \qquad (4.6a)$$

In table 4.1 we list the polynomials $P_m(z)$ for m = 3, 5 and 7.

Table 4.1. Optimal polynomials $P_m(z)$ for m = 3, 5 and 7

m = 3	$P_3(z) = 1 + z + \frac{1}{2} z^2 + \frac{1}{8} z^3$
m = 5	$P_5(z) = 1 + z + \frac{1}{2} z^2 + \frac{5}{32} z^3 + \frac{1}{32} z^4 + \frac{1}{64} z^5$
m = 7	$P_7(z) = 1 + z + \frac{1}{2} z^2 + \frac{35}{216} z^3 + \frac{1}{27} z^4 + \frac{14}{729} z^5 + \frac{1}{1458} z^6 + \frac{1}{2916} z^7.$

4.2. Almost second order formulas using two arrays of storage

In the derivation of the Runge-Kutta matrices M_j and N_{jl} we shall only consider the case that these matrices do not depend on the stepsize h. From the condition (3.5) for schemes only using two arrays of storage, together with the conditions (3.1) and (3.2), we find that $R_m(hJ_n)$ is given by

$$R_m(hJ_n) = \prod_{l=0}^{m-1} (I + hN_{l+1,l}J_n). \tag{4.7}$$

Writing $N_{l+1,l} = \begin{vmatrix} \mu_1 I & 0 \\ 0 & \beta_1 I \end{vmatrix}$, and substituting for T and J_n in the relation

$$T \begin{vmatrix} P_m(h\Lambda) & Q_m(h\Lambda) \\ Q_m(h\Lambda) & P_m(h\Lambda) \end{vmatrix} T^{-1} = R_m(hJ_n)$$

the matrix of eigenvectors and the Jacobian of the model problem (3.13), we obtain the stability conditions

$$p_k + q_k = \sum_{j=1}^{m} \mu_j \sum_{r=1}^{j-1} \beta_r \sum_{s=1}^{r-1} \mu_s \cdots \sum_{t=1} .$$

(k sums, last term is β for k even, else μ)

$$p_k - q_k = \sum_{j=1}^{m} \beta_j \sum_{r=1}^{j-1} \mu_r \sum_{s=1}^{r-1} \beta_s \cdots \sum_{t=1} .$$

(last term is μ for k even, else λ), $k = 3, \ldots, m$. (4.8)

Moreover, we derive from (4.1) the conditions for consistency of order two :

$$\sum_{j=1}^{m} \mu_j = 1, \quad \sum_{j=1}^{m} \beta_j = 1, \quad \sum_{j=2}^{m} \beta_j \sum_{k=1}^{j-1} \mu_k = \frac{1}{2},$$

$$\sum_{j=2}^{m} \mu_j \sum_{k=1}^{j-1} \beta_k = \frac{1}{2}, \quad \sum_{j=2}^{m} \mu_j \sum_{k=1}^{j-1} \mu_k = \frac{1}{2}. \tag{4.9}$$

Obviously, (4.8) and (4.9) consist of (2m + 1) equations in the (2m) unknowns β_j and μ_j, so that we may not expect to find a solution yielding a second order scheme with optimal polynomials $P_m(z)$ and $Q_m(z)$. Of course we could have found second order schemes by admitting less optimal P_m and Q_m. However, we did remove the last consistency condition, so that the schemes constructed are only of second order if $J_{11} = 0$ (e.g. which is the case by second order equations without first derivatives, written as a system of first order equations).
Now, we can easily calculate the parameters β_j and μ_j from (4.8) and (4.9) for polynomials P_m and Q_m given by (4.6a). However, a more efficient set of formulas is given by the relations

$$\mu_1 = \mu_m = \frac{1}{2k}, \quad \mu_{even} = \beta_{odd} = 0, \quad \mu_{odd} = \beta_{even} = \frac{1}{k},$$

$$m = 2k + 1. \tag{4.10}$$

It is easily verified that the first four relations of (4.9) are satisfied, whereas substitution in (4.8) yields for m = 3, 5 and 7 the polynomials

$$\tilde{P}_3(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{8}z^3, \quad \tilde{Q}_3(z) = \frac{1}{8}z^3,$$

$$\tilde{P}_5(z) = 1 + z + \frac{1}{2}z^2 + \frac{5}{32}z^3 + \frac{1}{32}z^4 + \frac{1}{256}z^5,$$

$$\tilde{Q}_5(z) = \frac{1}{32}z^3 + \frac{1}{256}z^5,$$

$$\tilde{P}_7(z) = 1 + z + \frac{1}{2}z^2 + \frac{35}{216}z^3 + \frac{1}{27}z^4 + \frac{1}{162}z^5 + \frac{1}{1458}z^6 + \frac{1}{17496}z^7,$$

$$\tilde{Q}_7(z) = \frac{1}{72}z^3 + \frac{1}{486}z^5 + \frac{1}{17496}z^7. \tag{4.11}$$

These polynomials are different from those listed in table 4.1 for m = 5 and m = 7; however, they satisfy the relations (4.4) and (4.6), so that the imaginary stability boundary β_{im} is again equal to 2 (m = 3), 4 (m = 5) and 6 (m = 7). The advantage of the formulas given by (4.10) lies in the fact that per Runge-Kutta step only (k + 1) evaluations of the first component of the right hand side of (1.1) are required, and k evaluations of the second component. Thus, the computational work is approximately m/2 right hand side evaluations. Defining *the effective stability boundary* $\beta_{eff,\,im}$ as the quotient of β_{im} and the number of right hand side evaluations per Runge-Kutta step, we obtain values as listed in table 4.2.

Table 4.2. β_{im}, $\beta_{eff,\,im}$ and the number of right hand side evaluations

m	Scheme generated by	β_{im}	number of r.h.s.eval.	$\beta_{eff,im}$
3	P_3 and Q_3 from table 4.1	2	$\leqslant 3$	$\geqslant .67$
3	$\tilde{P}_3$ and $\tilde{Q}_3$ from 4.11	2	1.5	1.33
5	P_5 and Q_5 from table 4.1	4	$\leqslant 5$	$\geqslant .80$
5	$\tilde{P}_5$ and $\tilde{Q}_5$ from 4.11	4	2.5	1.60
7	P_7 and Q_7 from table 4.1	6	$\leqslant 7$	$\geqslant .86$
7	$\tilde{P}_7$ and $\tilde{Q}_7$ from 4.11	6	3.5	1.71

From these results we expect that all schemes generated by (4.10) have, for odd values of m, a β_{im} equal to m−1.

For large values of m we would then obtain a $\beta_{eff,im}$ which is approximately equal to 2. However, we did not succeed in proving this relation for all odd m. Finally, we remark that the schemes generated by (4.10) look like the "symmetrized-scheme" proposed by VAN DER HOUWEN [4] for the integration of the shallow water equations. We intend to apply our schemes to these equations in the near future.

4.3. Second order formulas using three arrays of storage

When we allow ourselves three arrays of storage, it turns out to be possible to satisfy the conditions for second order consistency (4.1) and for stability (4.8). Thus, we shall try to reduce the number of right hand side evaluations per step. For that purpose we consider two subclasses of schemes defined by (3.7). First we choose $A_j = I$, $B_j = +I$, $D_j = 0$, $E_j = -F_j$, $j = 1, \ldots, m$, which yields for N_{jl} and M_j the relations

$$M_j = I, \quad N_{jl} = 0, \quad j = 1, \ldots, m, \quad l = 0, \ldots, j-2. \tag{4.12}$$

Writing $N_{j,j-1} = \begin{pmatrix} \mu_j I & 0 \\ 0 & \beta_j I \end{pmatrix}$, as in the previous section, we find the following relations for μ_j and β_j :

$$\mu_m = \beta_m = 1, \quad \mu_m \mu_{m-1} = \mu_m \beta_{m-1} = \beta_m \mu_{m-1} = \frac{1}{2},$$

$$p_k = (\mu_m \beta_{m-1} \mu_{m-2} \cdots \cdot_{m-k+1}$$

$$+ \beta_m \mu_{m-1} \beta_{m-2} \cdots \cdot_{m-k+1})/2$$

$$q_k = (\mu_m \beta_{m-1} \mu_{m-2} \cdots \cdot_{m-k+1}$$

$$- \beta_m \mu_{m-1} \beta_{m-2} \cdots \cdot_{m-k+1})/2. \tag{4.13}$$

For given p_k and q_k the parameters μ_j and β_j can be determined easily. The results, corresponding to the polynomials $\tilde{P}_m(z)$ and $P_m(z)$ as listed in formula (4.11) and table 4.1 are given below :

$$m = 3, \ N_{32} = I, \ N_{21} = \frac{1}{2} I, \ N_{10} = \begin{vmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{vmatrix}. \tag{4.14a}$$

$$m = 5, \ N_{54} = I, \ N_{43} = \frac{1}{2} I, \ N_{32} = \begin{vmatrix} \frac{3}{8} & 0 \\ 0 & \frac{1}{4} \end{vmatrix},$$

$$N_{21} = \begin{vmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{6} \end{vmatrix}, \quad N_{10} = \begin{vmatrix} \frac{1}{4} & 0 \\ 0 & 0 \end{vmatrix}. \tag{4.14b}$$

$$m = 7, \ N_{76} = I, \ N_{65} = \frac{1}{2} I, \ N_{54} = \frac{1}{54} \begin{vmatrix} 19 & 0 \\ 0 & 16 \end{vmatrix},$$

$$N_{43} = \frac{1}{76} \begin{vmatrix} 19 & 0 \\ 0 & 16 \end{vmatrix}, \quad N_{32} = \frac{1}{9} \begin{vmatrix} 2 & 0 \\ 0 & 1 \end{vmatrix},$$

$$N_{21} = \frac{1}{12} \begin{vmatrix} 2 & 0 \\ 0 & 1 \end{vmatrix}, \quad N_{10} = \frac{1}{6} \begin{vmatrix} 1 & 0 \\ 0 & 0 \end{vmatrix}. \tag{4.14c}$$

$$m = 5, \ N_{54} = I, \ N_{43} = \frac{1}{2} I, \ N_{32} = \begin{vmatrix} \frac{5}{8} & 0 \\ 0 & 0 \end{vmatrix},$$

$$N_{21} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{1}{5} \end{vmatrix}, \quad N_{10} = \begin{vmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{vmatrix}. \tag{4.14d}$$

$$m = 7, \ N_{76} = I, \ N_{65} = \frac{1}{2} I, \ N_{54} = \begin{vmatrix} \frac{35}{54} & 0 \\ 0 & 0 \end{vmatrix},$$

$$N_{43} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{8}{35} \end{vmatrix}, \quad N_{32} = \begin{vmatrix} \frac{14}{27} & 0 \\ 0 & 0 \end{vmatrix},$$

$$N_{21} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{1}{28} \end{vmatrix}, \quad N_{10} = \begin{vmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{vmatrix}. \tag{4.14e}$$

Obviously, the schemes (4.14d) and (4.14e) are more efficient than (4.14b) and (4.14d), as more zeros appear in the matrices. In table 4.3 we mention the effective stability boundary for these schemes.
When we try to maximize the number of zeros in the parameter matrices N_{jl}, (3.7) seems to impose a too severe condition. However, schemes requiring only $(m+1)/2$ right hand side evaluations can be constructed, when we consider the class of formulas given by

$$N_{jl} = 0, \quad j = 1, \ldots, m, \quad l = 1, \ldots, j-2, \text{ and } N_{m0} = 0,$$

$$M_j = I, \ j = 1, \ldots, m, \ N_{j0} = N_{10}, \ j = 2, \ldots, m-1. \tag{4.15}$$

The consistency conditions now read

$$\mu_m = \beta_m = 1, \ \mu_m(\mu_1 + \mu_{m-1}) = \mu_m(\beta_1 + \beta_{m-1})$$

$$= \beta_m(\mu_1 + \mu_{m-1}) = \frac{1}{2}$$

whereas the coefficients of the stability functions are given by

$$p_k + q_k = \mu_m \beta_{m-1} \cdots (\cdot_1 + \cdot_{m-k+1}) ;$$

(. stands for μ, if k is odd, else β)

$$p_k - q_k = \beta_m \mu_{m-1} \cdots (\cdot_1 + \cdot_{m-k+1}),$$

(. stands for β if k is odd, else μ).

Choosing the coefficients p_k and q_k as given in formula

(4.11), we obtain the following schemes :

$$m = 3,\quad N_{32} = I,\quad N_{21} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{1}{2} \end{vmatrix},$$

$$N_{20} = N_{10} = \begin{vmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{vmatrix}. \qquad (4.16a)$$

$$m = 5,\quad N_{54} = I,\quad N_{43} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{1}{2} \end{vmatrix},\quad N_{32} = \begin{vmatrix} \frac{1}{8} & 0 \\ 0 & 0 \end{vmatrix},$$

$$N_{21} = \begin{vmatrix} 0 & 0 \\ 0 & 1 \end{vmatrix},$$

$$N_{40} = N_{30} = N_{20} = N_{10} = \begin{vmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{vmatrix}. \qquad (4.16b)$$

$$m = 7,\quad N_{76} = I,\quad N_{65} = \begin{vmatrix} 0 & 0 \\ 0 & \frac{1}{2} \end{vmatrix},\quad N_{54} = \begin{vmatrix} \frac{4}{27} & 0 \\ 0 & 0 \end{vmatrix},$$

$$N_{43} = \begin{vmatrix} 0 & 0 \\ 0 & 1 \end{vmatrix},\quad N_{32} = \begin{vmatrix} \frac{1}{54} & 0 \\ 0 & 0 \end{vmatrix},$$

$$N_{21} = \begin{vmatrix} 0 & 0 \\ 0 & 1 \end{vmatrix},$$

$$N_{60} = N_{50} = N_{40} = N_{30} = N_{20} = N_{10} = \begin{vmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{vmatrix}. \qquad (4.16c)$$

The effective stability boundaries of these schemes are given in table 4.3.
As the matrices N_{jl} are sparse, implementation of these schemes using not more than three arrays is possible, too.

Table 4.3. The effective stability boundary of the schemes (4.14) and (4.16)

m	Scheme	β_{im}	number of r.h.s. eval.	$\beta_{eff,im}$
3	4.14a	2	$2\frac{1}{2}$	.80
3	4.16a	2	2	1.00
5	4.14b	4	$4\frac{1}{2}$	.89
5	4.14d	4	$3\frac{1}{2}$	1.14
5	4.16b	4	3	1.33
7	4.14c	6	$6\frac{1}{2}$	.92
7	4.14e	6	$4\frac{1}{2}$	1.33
7	4.16c	6	4	1.67

Remark

When we apply the schemes determined by (4.10) to the second order equation $\frac{d^2y}{dx^2} = g(y)$, we obtain the relations :

(a) $m = 3$: $y_{n+1} = y_n + hy_n' + \frac{1}{2}h^2 g(y_n + \frac{1}{2}hy_n')$,

$$y_{n+1}' = 2\frac{y_{n+1} - y_n}{h} - y_n',$$

(b) $m = 5$: $y_{n+1}^{(1)} = y_n + \frac{1}{4}h\,y_n'$,

$$y_{n+1}^{(2)} = y_{n+1}^{(1)} + \frac{1}{2}hy' + \frac{1}{4}h^2 g(y_{n+1}^{(1)})$$

$$y_{n+1} = y_n + hy_n' + \frac{3}{8}h^2 g(y_{n+1}^{(1)}) + \frac{1}{8}h^2 g(y_{n+1}^{(2)}),$$

$$y_{n+1}' = y_n' + \frac{1}{2}hg(y_{n+1}^{(1)}) + \frac{1}{2}hg(y_{n+1}^{(2)}).$$

As these formulas require only two arrays of storage, they are more economical than formula (2.5), which requires three arrays. We mention that this formula, which was devised for second order equations without first derivatives in VAN DER HOUWEN [5], can be constructed by the methods described in this report, too. In fact, let us consider almost second order formulas (the condition $\Sigma_j N_{ml}(p, q)\,\Sigma\, N_{lk}(r, s) = \frac{1}{2}\delta_{pq}\delta_{rs}$ need not be satisfied for $r = s = p = q = 1$), which use three arrays of storage. Setting

$$N_{54} = \begin{vmatrix} I & 0 \\ 0 & 0 \end{vmatrix} \text{ and } N_{53} = \begin{vmatrix} 0 & 0 \\ 0 & I \end{vmatrix},$$

and using $\tilde{P}_5(z)$ and $\tilde{Q}_5(z)$ from (4.11), we obtain formula (2.5) by a relation similar to (4.13).

4.4. Strongly stable formulas

The formulas generated in the preceding sections are only weakly stable, as their associated polynomials $P(z)$ and $Q(z)$ satisfy (3.16) with the equality sign. Indeed, it is easily verified that their amplification factors α are exactly in modulus 1. Strongly stable formulas, whose amplification factors are bounded by a damping function $\sqrt{\rho(z)}$, can be constructed as

Table 4.4. Runge-Kutta parameters for second order strongly stable schemes

Associated polynomials	Damping function	$\beta = \beta_{im}$	RK parameters	
$P_{3,\epsilon}(z)$	$1 - \frac{\epsilon z^4}{\beta^4}$	$\sqrt{4-2\epsilon}$	$\mu_3 = 1$	$\beta_3 = 1$
			$\mu_2 = \frac{1}{2}$	$\beta_2 = \frac{1}{2}$
			$\mu_1 = \frac{1}{2} + \frac{2\epsilon}{\beta^4}$	$\beta_1 = 0$
$P_{5,\epsilon}(z)$	$1 - \frac{\epsilon z^4}{\beta^4}$	$\sqrt{8(1+\sqrt{1-\epsilon})}$	$\mu_5 = 1$	$\beta_5 = 1$
			$\mu_4 = \frac{1}{2}$	$\beta_4 = \frac{1}{2}$
			$\mu_3 = \frac{1}{2} + \frac{2\beta^2 - 2\epsilon}{\beta^4}$	$\beta_3 = 0$
			$\mu_2 = 0$	$\beta_2 = \frac{4\beta^2 - 8\epsilon}{\beta^4 + 4\beta^2 - 4\epsilon}$
			$\mu_1 = \frac{1}{2}$	$\beta_1 = 0$
$P_{7,\epsilon}(z)$	$1 - \frac{3\epsilon z^4}{\beta^4} + \frac{2\epsilon z^6}{\beta^6}$	$\sqrt{36-9\epsilon}$	$\mu_7 = 1$	$\beta_7 = 1$
			$\mu_6 = \frac{1}{2}$	$\beta_6 = \frac{1}{2}$
			$\mu_5 = \frac{35+\epsilon}{54}$	$\beta_5 = 0$
			$\mu_4 = 0$	$\beta_4 = \frac{32+3\epsilon}{140+4\epsilon}$
			$\mu_3 = \frac{1}{27}\frac{448+45\epsilon}{32+3\epsilon}$	$\beta_3 = 0$
			$\mu_2 = 0$	$\beta_2 = \frac{16+4\epsilon}{448+45\epsilon}$
			$\mu_1 = \frac{1}{2}$	$\beta_1 = 0$

described by VAN DER HOUWEN [5].
Instead of the conditions (3.16) we now satisfy

$$|P(z)|^2 - |Q(z)|^2 \leq \rho(z)$$
$$\text{and} \qquad (4.17)$$
$$|\mathrm{Re}\, P(z)| \leq \rho(z) .$$

Setting again (cf. (4.6)) $Q(z) = P(z) - 1 - z - \frac{1}{2}z^2$, the derivation of Re P(z) is completely analogous to the derivation of S(z) in [5]. Therefore let it be sufficient to give the resulting polynomials, together with their β_{im}, for m = 3, 5 and 7.

$$P_{3,\epsilon}(z) = 1 + z + \frac{1}{2}z^2 + (\frac{1}{8} + \frac{\epsilon}{2\beta^4})z^3 ,$$

$$\beta_{im} = \beta = \sqrt{4-2\epsilon} , \qquad (4.18)$$

The related damping function is $\rho(z) = 1 - \frac{\epsilon z^4}{\beta^4}$.

$$P_{5,\epsilon}(z) = 1 + z + \frac{1}{2}z^2 + (\frac{1}{8} + \frac{\epsilon}{2\beta^4} + \frac{\beta^2 - 2\epsilon}{2\beta^4})z^3$$
$$+ \frac{\beta^2 - 2\epsilon}{2\beta^4} + \frac{\beta^2 - 2\epsilon}{4\beta^4} z^5 ,$$

$$\beta_{im} = \beta = \sqrt{8(1+\sqrt{1-\epsilon})}. \qquad (4.19)$$

Again, the related damping function is $\rho(z) = 1 - \frac{\epsilon z^4}{\beta^4}$.

$$P_{7,\epsilon}(z) = 1 + z + \frac{1}{2} z^2 + \left(\frac{35}{216} + \frac{\epsilon}{216}\right) z^3 + \left(\frac{1}{27} + \frac{\epsilon}{288}\right) z^4 + \left(\frac{14}{729} + \frac{5\epsilon}{2592}\right) z^5 + \left(\frac{4+\epsilon}{5832}\right) z^6 + \frac{1}{2}\left(\frac{4+\epsilon}{5832}\right) z^7,$$

$$\beta_{im} = \beta \approx 6 - \frac{3}{4}\epsilon$$

(terms of order ϵ^2 are neglected).

The damping function related to $P_{7,\epsilon}(z)$ is

$$\rho(z) = 1 - \frac{3\epsilon}{\beta^4} z^4 - \frac{2\epsilon z^6}{\beta^6}.$$

Now, using the above values of the coefficients p_k and q_k of the polynomials $P(z)$ and $Q(z)$, we can compute the Runge-Kutta parameters μ_j and β_j by means of (4.8) or (4.13). As the schemes computed by using (4.8) contain only few zeros compared to the schemes (4.10), which are exactly the same ones for $\epsilon = 0$, the use of a damping substitute for (4.10) does not seem appropriate.
However, using (4.13) for the calculation of the μ_j and β_j, we find only slight modifications of (4.14a), (4.14d) and (4.14e). The resulting parameter values are listed in table 4.4.

REFERENCES

1. FEHLBERG, E. : "Classical eight- and lower-order Runge-Kutta-Nyström formulas with step-size control for special second-order differential equations", Technical Report, NASA TR R-381, Marshall Space Flight Centre, Alabama, 1972.

2. VAN DER HOUWEN, P. J. : "Stabilized Runge-Kutta methods with limited storage requirements", Report TW 124/71, Mathematisch Centrum, Amsterdam, 1971.

3. VAN DER HOUWEN, P. J. : "Explicit Runge-Kutta formulas with increased stability boundaries", Numer. Math. 20 (1972), pp. 149-164.

4. VAN DER HOUWEN, P. J., : "Two-level difference schemes with varying mesh sizes for the shallow water equations", Report NW 22/75, Mathematisch Centrum, Amsterdam, Amsterdam 1975.

5. VAN DER HOUWEN, P. J. : "Stabilized Runge-Kutta methods for second-order differential equations without first derivatives", Report NW 26/75, Mathematisch Centrum, 1975.

6. KREIS, H., and OLIGER, J. : "Methods for the approximate solution of time dependent problems", GARP publication series no 10, Geneva, 1973.

7. LAMBERT, J. D. : "Computational methods in ordinary differential equations", John Wiley & Sons, London, 1973.

8. ZONNEVELD, J. A. : "Automatic numerical integration", MC Tract 8, Mathematisch Centrum, Amsterdam, 1964.

Formula manipulation in ALGOL 68 and application to Routh's algorithm

Abstract. This paper presents an implementation of Routh's algorithm and the Schur criterion, in order to determine the location of the zeros of a complex polynomial in one variable, over the ring of multivariate polynomials with integral coefficients. Both algorithms are applied to the characteristic polynomials of multistep methods. Moreover, procedures are given for the arithmetic operations +, -, *, ÷ with arbitrarily long integral numbers as operands.

Formula manipulation in ALGOL 68 and application to Routh's algorithm

Introduction

The determination of the location of the zeros of a polynomial is important in the study of stability of difference equations. There is a vast amount of literature on this topic (see for example Marden) and several algorithms are known, giving conditions for a number of zeros to lie in a region of the complex plane (Barnett & Siljak). Two regions are of particular interest in the stability analysis: the unit circle and the left-half plane. An algorithm, sometimes called the Schur criterion (Lambert, Miller), determines whether or not all zeros lie in the unit circle, or on the boundary. By means of the Routh array the number of zeros in the left-half plane (and on the imaginary axis) can be calculated.
In this paper we give an implementation of these algorithms for polynomials over the ring of multivariate polynomials with integral coefficients. We choose ALGOL 68 (Van Wijngaarden) as our implementation language, because it permits the definition of new data types and operations on elements of these types.

The implementation consists of three separate parts. First, we develop operations for long integer arithmetic. These operations make use of TORRIX, a system for operations on vectors and matrices, written by Van der Meulen and Veldhorst. The second part is a system for manipulation with polynomials over a ring, being a subset of a package of Teer. The third part is formed by routines for the implementation of the Schur criterion and the creation of the Routh array. We note, that the three parts are independant and each one can be used alone, although they are meant to be used in combination with each other.

Finally, we give a few simple examples. In a second paper we will apply our system to the characteristic polynomials of linear multistep methods for first and second order differential equations.

1. *Long integer arithmetic*

The domain of integers in a computer representation is usually bounded. Let *maxint* be a value, such that the operations + , - and * are correctly performed, when both operands and result lie in the range [-maxint, maxint]. Let *radix* be an arbitrary positive integer,

such that

(1) $\quad 1 < radix^2 \leq maxint$.

Then, we can represent each integer with an arbitrary number of digits by a polynomial in *radix*

(2) $\quad \sum_{j=0}^{k} a_j \; radix^j$;

this representation is unique (except for possible leading zeros) when the coefficients satisfy

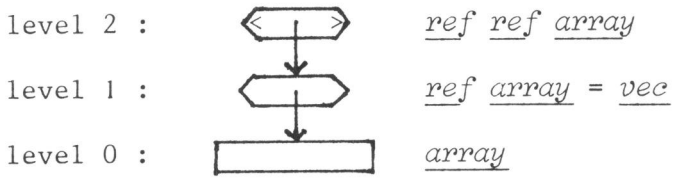
(3) $\quad |a_j| < radix, \quad j=0,\ldots,k$,

$\quad a_i \; a_j \geq 0 \quad ,i,j=0,\ldots,k$.

Under the conditions (1) and (3) we have exact arithmetic for operations on the coefficients of these polynomials. We may thus simulate exact long integer arithmetic by defining the operations +,-,÷,* for polynomials of form (2). As data-type for these polynomials we have chosen the TORRIX-vector (Van der Meulen & Veldhorst). In the definition of the operations TORRIX plays an important role, too; therefore we mention at first some important features of this programming system.

Torrix

A vector in the TORRIX-system is an array of arbitrary length, consisting of a concrete and a virtual part. All elements of the virtual part are assumed to be zero; the concrete part, usually containing non-zero elements, is represented in computer memory. The addition of two vectors is always defined, whether the bounds of the concrete parts coincide or not. Of course, the resulting vector has a concrete part of minimal length, in order to save memory space. In many operations new arrays are generated on a heap; because they have varying lifetimes, a good garbage collector is required. The level structure is another feature of TORRIX: on the lowest level we change values of elements of concrete arrays, on a higher level the references to these arrays are changed. The relation between the various levels is visualized by a picture (Van der Meulen, page 69):



Picture 1. The three TORRIX-levels.

Finally, we mention that TORRIX allows freedom in the type of the array elements. For our purpose, we have integral elements only. Moreover, we will only need a part of the operations defined on vectors, and none of the operations on matrices. We list them all in section 4.

Representation

For the representation of long integers we have chosen the data structure for vectors as described in TORRIX. A long integer is a *ref array* variable; the contents of the array can be changed, but also the reference. Thus, shrinking and expanding of long integers can be handled. However, for the practical implementation we have a slightly different definition of a long integer:

(4) *mode longint* = *struct* (*vec a*) .

The introduction of this rather trivial structure is necessary, because the operations +,-,etc. on *vec*s are already defined in TORRIX, and we will define them for *longint*s in a different way, which requires that *vec* and *longint* are different modes. The alternatives, rewriting TORRIX or giving new symbols to our operations, seem unattractive to us, because much of the clarity of the source texts would vanish. However, for an efficient implementation, the first alternative might be chosen, especially when the TORRIX part is hidden from the user.
A special element in the system is called *longzero*, a structure containing *zerovec*. This element represents the zero of the long integer system, and is used in testing against zero.

Operations

The result of a TORRIX operation is usually a vector which does not satisfy the conditions (3). Thus, we have to normalise the resulting vector, in order to obtain the unique representation:

(5) *proc normalise* = (*ref vec v*) *ref vec* : (..)

In this procedure it is assumed that all elements of the array v are in absolute value ≤ *radix* (*radix*-1) ; moreover, the most significant non-zero element of v should determine the sign of the result, if this result is not equal to zero. This is a realistic assumption. After a substraction of two normalised numbers the elements of v may have different signs, but the sign of the result is equal to the sign of the most significant non-zero element. Moreover, after a multiplication and an addition we have

(6) $|a * b + c| \leq radix * (radix-1)$,

if $|a|$, $|b|$ and $|c|$ are less than *radix* .

Now,addition and subtraction are performed by the TORRIX operation, followed by a normalisation.

We note, that these operations can be made more efficient by postponing the normalisations until they are unavoidable (e.g. before a multiplication). Multiplication can be performed by forming the Cauchy-product of the arrays; however, in the computation of the product sums $\Sigma\, a_i\, b_{j-i}$ integer overflow may occur, and care should be taken. For that end we have rewritten the corresponding TORRIX routine. We remark that a considerable gain in efficiency can be obtained by using the TORRIX operator for the Cauchy product. The result will be correct under the restriction that at least one operand is less in absolute value than $radix^n$, where

(7) $(n+1)\ radix^2 \leq maxint$.

In our implementation, make a test and if this inequality is satisfied we perform fast multiplication.

Exponentiation is defined by repeated multiplication, using the relation $a^{2k+1} = (a^2)^k \,.\, a$ (e.g. Wirth, page 29).

The integer division $u \div v$ is calculated in several steps. At first, both operands are made positive, and multiplied by a factor, such that

(8) $radix \div 2 \leq v[upb\ v] < radix$.

Let $n = upb\ u - upb\ v$, and $u^{(n)} = u$. Then successively integers q_i , $i = n, n-1, \ldots, 0$, are calculated, such that the relations

(9) $$0 \le u^{(i-1)} = u^{(i)} - q_i\ v\ radix^i < v\ radix^i \ , \quad 0 \le i \le n \ ,$$

hold; after multiplication with a sign this yields the coefficients of the quotient. The greatest common divisor is calculated by means of Euclid's algorithm.
The relational operations are rather trivial, and we do not mention them explicitly. They are all listed in section 4. Finally, we remark that all operations are defined also for *longint* and *int* operands by means of the operator

(10) *op widen* = (*int i*) *longint* : (...) ;

which converts an integer (possibly larger than *radix*) into a long integer.

2. *Polynomial manipulation*

In this section we will shortly describe operations for manipulation with multivariate polynomials. Most of these operations are given in (Teer), to whom we refer for a more detailed description.
A polynomial in n-variables, say $x_1,\ldots,x_n$, can be regarded as a polynomial in x_n , whose coefficients are polynomials in the remaining variables, and so on. The order in which the variables are selected, can be any. We have chosen a lexicographical ordering, such that xy^2z is regarded as a polynomial in x . Finally, we obtain polynomials in one variable, with as coefficients elements of some algebraic system, e.g. the ring of integral numbers. The polynomial coefficients are represented as a linked list, which will be more efficient than an array, when we deal with sparse polynomials (Knuth, page 251):

(1) *mode formula* = *union* (*ref const*, *ref pol*) ;

(2) *mode pol* = *struct* (*string variable*, *ref llterms next*) ;

(3) *mode llterms* = *struct* (*ref llterms next*, *formula coef*, *int exp*) .

For the *mode const* we have chosen

(4) *mode const = struct (longint value)* ;

the structure is necessary for reasons analogous to those in the previous section with respect to the definition of the *mode longint* ; the operators to be defined on *formulas* are already defined for *longints*. We have two neutral elements, for multiplication and addition, given by

(5) *const zero = (const i ; value of i := longzero ; i)* ;

(6) *const one = (const i ; value of i := widen 1 ; i)* .

Remark These definitions cannot be simplified to for instance *const zero = const (longzero)* , because *(longzero)* is not a *struct* and a cast for a *struct* with one field is not allowed in ALGOL 68.
The operations defined between *formulas* may be defined for an integer operand, too. To this end we declare

(7) *proc makenumber = (int i) ref const : (...)* ,

which converts an *int* into a *const*. We also have a widening from type *string* to *formula*:

(8) *proc algvar = (string s) ref pol : ( )* ,

which generates a polynomial of degree 2 in *s*, with a constant coefficient 1.

Remark The operations defined are slightly different from the original version given by Teer. In Teer's version the constant coefficients have *mode int* , which may be more efficient, but leads rather often to overflow in our applications. A minor change, involving only the procedure *makenumber* and the constants *one* and *zero*, made the system applicable to polynomials with *longint* coefficients. Moreover, we added the operator

(9) *op gcc = (formula f) ref const : (..)* ,

which calculates the greatest common divisor of the constant coefficients of the polynomial *f* . Division of *f* by the obtained value will reduce the size of the integral coefficients considerably in many cases.

The elimination of rational functions is a more drastic deviation from the original system. In all operations, we cut out the parts in which rational functions were involved; in the applications of the next section we deal with polynomials only, and division is applied only when it is known a priori that there will be no remainder.

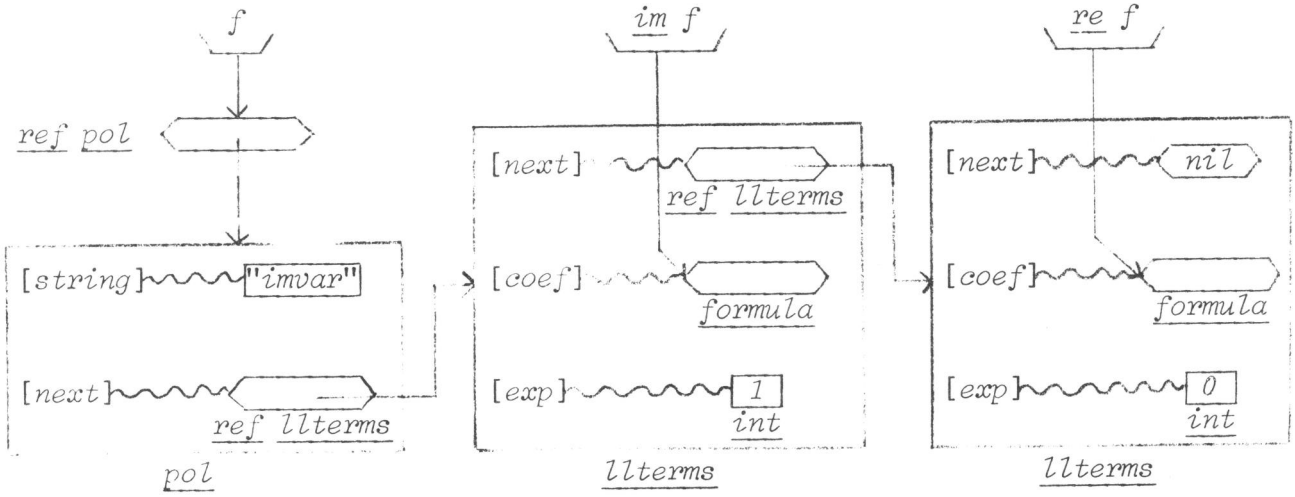
Complexification

We may regard a polynomial with complex (integral) coefficients as a polynomial in one more variable, with integral coefficients. In this way, we simulate operations on complex polynomials without introducing another data-structure. Moreover, by choosing the "imaginary variable" in such a way, that it dominates all other variables, the selections

(10) *op re* = (*formula f*) *formula* : (..) ;

(11) *op im* = (*formula f*) *formula* : (..)

are performed very efficiently, which is illustrated in the picture:



Had we defined our coefficients as complex numbers, the selections *re* and *im* would have been very laborious, as we would have had to go down to the lowest level and create new lists and copies. Moreover, each operation would involve complex arithmetic, whether complex numbers are used or not. In our system, we do not have to bother about complex numbers, as long as they are not involved. The price we have to pay for this feature is that the representation of a complex polynomial is not unique: for example a multiplication of two polynomials may yield a polynomial in

"imvar" of degree 2. In order to reduce a polynomial in *"imvar"* to its standard form (i.e. degree less than 2), we defined

(12) *op complex* = (*formula f*) *formula* : (..) .

Substitution

One may wish to replace a variable of a polynomial by an expression (a constant or a polynomial). To that end we supply

(13) *proc substitute* (*string s*, *formula fs*, *f*) *formula* : (..) ,

which generates a new *formula*, replacing all occurences of the variable s in the *formula f* by the *formula fs*.

3. *Routh's algorithm*

The Routh-Hurwitz criterion gives conditions for a real polynomial to have roots in the left half plane. A well-known formulation is given in e.g. (Lambert, page 81) and (Marden, page 181). The roots of a real polynomial $P_n(x) = p_0x^n + p_1x^{n-1} + \ldots + p_n$ $(p_0>0)$ lie in the left half plane if and only if the matrices Q_k given by

$$(1) \qquad Q_k = \begin{pmatrix} p_0 & p_2 & p_4 & \cdots & p_{2k} \\ 0 & p_1 & p_3 & \cdots & p_{2k-1} \\ 0 & p_0 & p_2 & \cdots & p_{2k-2} \\ 0 & 0 & p_1 & \cdots & p_{2k-3} \\ 0 & 0 & 0 & \cdots & p_k \end{pmatrix} , \quad (p_j = 0, j>n)$$

have positive determinants, $k = 0,\ldots,n$.
It can be shown that this condition implies the positivity of the coefficients of p_n ; thus

$$(2) \qquad p_j > 0 \ , \ j = 0,\ldots,n \ ,$$

are necessary conditions. The conditions $\det(Q_k) > 0$ involve the calculation of determinants, and these can be rather complicated.
An equivalent formulation, yielding as straight-forward an algorithm, is given by (Barnett and Siljak, 3.16).

Let the rows $r_0, r_1, \ldots$, be defined by ($r_i^{(j)}$ denotes the j-th element of row r_i)

$$
\begin{aligned}
r_0 &= p_0, p_2, p_4, \cdots \\
r_1 &= p_1, p_3, p_5, \cdots \\
r_{i+1}^{(j)} &= r_i^{(0)}\, r_{i-1}^{(j+1)} - r_{i-1}^{(0)}\, r_i^{(j+1)} \quad , \quad i = 1, \ldots, n-1 \ ,
\end{aligned} \tag{3}
$$

then the roots of p_n lie in the left half plane, if and only if

$$
r_i^{(0)} > 0 \ , \ i = 0, \ldots, n \ . \tag{4}
$$

When the coefficients of the polynomial are not simple constants, the recurrence relation (3) has a serious drawback: the number of coefficients of P_n , occuring in the terms of $r_k^{(j)}$, increases rapidly with k , like the Fibonacci numbers. We will show that (3) can be modified in such a way that we have only a linear increase.

In (Bareiss) a similar technique has been developed for the solution of linear equations by integer-preserving Gaussian elimination. To this end we observe that the rows $r_k, k=0, \ldots, n$, given by (3) , can be obtained from the matrix Q_n by repeated row-elimination. To be more specific, define a sequence of matrices $Q_k^{(i)}$ by

$$
Q_k^{(i+1)} = \begin{pmatrix} I_{i+1} & \vdots & 0 \\ \cdots & \vdots & \cdots \\ & L_i & \\ 0 & \vdots & \\ & L_i & 0 \\ & 0 & \ddots \end{pmatrix} Q_k^{(i)} \ , \quad \begin{matrix} k=0, \ldots, n \ , \\ i=0, \ldots, k-1 \ , \end{matrix} \tag{5}
$$

where $L_i = \begin{pmatrix} 1 & 0 \\ -t_i & t_{i+1} \end{pmatrix}$, $t_i = r_i^{(0)}$ and $Q_k^{(0)} = Q_k$.

Then, it is easily verified, that the first i rows of $Q_n^{(i)}$ are equal to the (shifted) rows $r_0, \ldots, r_{i-1}$, and the last rows of $Q_n^{(i)}$ are copies of r_i and r_{i+1} . Now, we consider the determinants of $Q_k^{(k)}$.

We have from (5) :

$$\det(Q_k^{(i+1)}) = \det(Q_k^{(i)}) \cdot t_{i+1}^{(k-i)\div 2} \quad , \; i = 0,\ldots, k-1 \; ,$$

and thus

$$(6) \qquad \det(Q_k^{(k)}) = \det(Q_k) \prod_{i=1}^{k-1} t_i^{(k+1-i)\div 2} \quad , \; 0 \le k \le n \; .$$

Moreover, $Q_k^{(k)}$ is upper triangular, with $t_0,\ldots,t_k$ on the main-diagonal; thus we have

$$(7) \qquad \prod_{i=0}^{k} t_i = \det(Q_k^{(k)}) = \det(Q_k) \cdot \prod_{i=1}^{k-1} t_i^{(k+1-i)\div 2} \quad , \; 0 \le k \le n \; ,$$

which implies that t_k can be divided by $\prod_{i=1}^{k-3} t_i^{(k-1-i)\div 2}$.

Using the same argument, replacing the last column of $Q_k^{(0)}$ by column $(k+1+j)$ of Q_n , it can be verified that $r_k^{(j)}$ can be divided by this product. The divisions should not of course be performed afterwards, but during the recursion, and this leads to the algorithm

$$(3^*) \qquad \begin{aligned} \tilde{r}_0^{} &= p_0, p_2, p_4, \ldots . \\ \tilde{r}_1^{} &= p_1, p_3, p_5, \ldots . \\ \tilde{r}_{i+1}^{(j)} &= (\tilde{t}_i \; \tilde{r}_{i-1}^{(j+1)} - \tilde{t}_{i-1} \; \tilde{r}_i^{(j+1)}) \; / \; \tilde{t}_{i-2} \quad , \; i \ge 1 \; , \\ \tilde{t}_0^{} &= \tilde{t}_{-1} = 1, \; \tilde{t}_i = \tilde{r}_i^{(0)} \quad , \; i \ge 1 \; . \end{aligned}$$

It is easily verified that the relation between $\tilde{r}_i^{(j)}$ and $r_i^{(j)}$ is indeed given by

$$(8) \qquad r_i^{(j)} = \tilde{r}_i^{(j)} \prod_{k=1}^{i-3} t_k^{(i-1-k)\div 2} \quad ,$$

Hence the conditions (4) are equivalent to

$$(4^*) \qquad \tilde{r}_i^{(0)} > 0 \quad , \; i = 0,\ldots, n \; .$$

Schur criterion

The Schur criterion, as it is called by Lambert, is another method for determining the location of the zeros of a (real or complex) polynomial.

Miller gives an extensive treatise of this criterion in his paper. We will summarize some of his results, with respect to the location of roots within or on the unit circle.

Let p_n be a complex polynomial of degree n ($n \geq 1$). Then, we define the associated polynomial p_n^* and the reduced polynomial p_{n-1} of degree $n-1$ by

$$p_n^*(x) = x^n \bar{p}_n(\frac{1}{x}) , \tag{9}$$

$$p_{n-1}(x) = \frac{p_n^*(0)\, p_n(x) - p_n(0)\, p_n^*(x)}{x}$$

We call a polynomial a *Schur polynomial* if all its roots lie in the open unit circle, and a *Von Neumann polynomial* if all roots lie in or on the unit circle. Moreover, a Von Neumann polynomial is called simple, if all its roots lying on the unit circle are simple. Miller states the following theorems:

Theorem 1: p_n is a Schur polynomial iff

$$|p_n^*(0)| - |p_n(0)| > 0 \tag{10}$$

and p_{n-1} is a Schur polynomial.

Theorem 2: p_n is a simple Von Neumann polynomial iff either (10) holds and p_{n-1} is a simple Von Neumann polynomial or $p_{n-1} \equiv 0$ and p_n' is a Schur polynomial.

We note that $p_{n-1} \equiv 0$ implies the equality of $|p_n^*(0)|$ and $|p_n(0)|$. (Consider the leading term x^{n-1}).

By a recursive application of these theorems a set of conditions (10) are obtained, for determining whether or not a polynomial is a Schur polynomial or a simple Von Neumann polynomial. However, for high degrees these conditions become rather complicated: each term of p_{n-k} will contain 2^k coefficients of the original polynomial, so that the complexity increases exponentially with the degree. For this reason, the Schur criterion will not be very useful for polynomials with complicated coefficients and degree higher than 4 or 5. However, for complex polynomials of low degree, the Schur criterion is more attractive than Routh's algorithm: in the latter algorithm we should form the real polynomial

$P_n(x)$ $\bar{P}_n(x)$ first, thus doubling the degree, before applying the recurrence relations (3^*) to this new polynomial.
Finally, we remark that determination of roots within the unit circle and within the left half plane are related problems. Let f_n be a Schur polynomial. Then, a simple transformation yields a polynomial g_n having all its roots in the left half plane:

$$(11) \qquad g_n(\xi) = (1-\xi)^n f_n(\zeta) , \quad \zeta = \frac{1+\xi}{1-\xi} .$$

The inverse transformation is given by

$$(12) \qquad f_n(\zeta) = (\zeta+1)^n g_n(\zeta) , \quad \xi = \frac{\zeta-1}{\zeta+1}$$

In section 4 we give procedures which calculate the conditions (4^*), the conditions (10) for $i = 1,\ldots,n$, and which perform the transformations (11) and (12). Moreover, we provide procedures in order to obtain the coefficients of a polynomial, and to construct a polynomial from its coefficients.

4. *Declarations and operators*

A. Torrix operations

A1 *proc torrix = (bool fatalerror, [ ] char message) void : (...);*
handles error-messages which may be warnings or fatal-errors.

op max = (int m,n) int : (..) ;
yields the maximum of *m* and *n* .

A2
```
mode scal     = int ;
mode array 1  = [mindex : maxdex] scal ;
mode vec      = ref array 1 ;
vec  zerovec  = heap [maxdex : mindex] scal ;
```

The first declaration defines the type of the *array* elements to be integral. *Mindex* and *maxdex* are the (implementation-dependent) virtual lower- and upperbounds of all *arrays*. We set *mindex* equal to zero by a call of *setgendex*.

A3 *proc setgendex = (int lower,upper) void : (..)*
sets particular values for *mindex* and *maxdex* .

proc genarray 1 = (int lwb,upb) vec : (..)
generates an *array* with concrete bounds *lwb* and *upb* and returns the *vec* referring to the newly created *array*.

A4	operator	prio	left operand	right operand	result
1	?	9	*vec*	*int*	*scal*
2	*copy*	10		*vec*	*vec*
3	*lwb,upb*	10		*vec*	*int*
4	*zero*	10		*vec*	*bool*
5	= /=	4	*vec*	*vec*	*bool*
6	*into*	2	*scal*	*vec*	*vec*
7	*trim*	10		*ref vec*	*ref vec*
8	*<*	1	*vec*	*scal*	*vec*
9	+	6	*vec*	*vec*	*vec*
10	-	6	*vec*	*vec*	*vec*
11	-	10		*vec*	*vec*
12	+:=	1	*ref vec*	*vec*	*ref vec*
13	-:=	1	*ref vec*	*vec*	*ref vec*
14	××	8	*vec*	*vec*	*vec*

1.	*u ? i*	returns zero if *u[i]* lies in the virtual part of *u* , else *u[i]* .
2.	*copy u*	generates a copy of the *array of u* .
3.	*lwb u*	returns the concrete lowerbound of *u* .
	upb u	returns the concrete upperbound of *u* .
4.	*zero u*	returns *true* when *u is zerovec*.
5.	*u = v*	returns *true* when all elements of the total-*array* s of *u* and *v* are equal .
	u /= v	is equivalent to *not* (*u=v*) .
6.	s *into u*	assigns the value s to all elements of the *array of u* .
7.	*trim u*	constructs a new descriptor in order to achieve that (u[*lwb u*]/=0) *and* (*u*[*upb u*]/=0) .
8.	*u *< s*	multiplies all elements of the *array of u* with *s*
9.	*u+v*	generates a new concrete *array* and assigns to its elements the sum $u_i + v_i$.
10.	*u-v*	generates a new concrete *array* and assigns to its elements the differences $u_i - v_i$.
11	*-u*	is equivalent to *zerovec - u* .
12	*u +:= v*	is, in its result, equivalent to *u := u+v* ; however, a new *array* is not generated when the result fits in *u* .
13.	*u -:= v*	is, in its result, equivalent to *u := u-v* ; however, a new *array* is not generated when the result fits in *u*.
14.	*u ×× v*	the convolution (Cauchy-) product of *u* and *v* .

B. LONGINT operations

B1 Fundamental declarations

```
mode longint      = struct (vec a) ;
longint longzero  = (longint c ; a of c := zerovec ; c) ;
int radix         = # a constant much larger than zero, such that
                    radix^2 ≤ maxint # ;
```

```
int length      =  # a constant such that radix ↑ (length+1) > maxint,
                   to be used in the operator widen # ;

op widen        =  (int i) longint : ( .. ) ;

    # the widening from integer to long integer #

proc normalise =  (ref vec v)  ref vec  :  ( .. ) ;

    # normalization of  v , such that  abs(v[i]) < radix  and
                                       v[i] * v[j] ≥ 0 #

proc genlongint=  (vec v) longint :  ( .. ) ;

    # generates a longint on the heap, with a field referring to  v #
```

B2 Operators

	operator	prio	left operand	right operand	result
1	*copy*	10		*longint*	*longint*
2	*abs*	10		*longint*	*longint*
3	*zero*	10		*longint*	*bool*
4	=	4	*longint*	*longint*	*bool*
	/=		*longint*	*int*	*bool*
			int	*longint*	*bool*
5	<	5	*longint*	*longint*	*bool*
	>		*longint*	*int*	*bool*
	<=		*int*	*longint*	*bool*
	>=				
6	+	6	*longint*	*longint*	*longint*
	-		*longint*	*int*	*longint*
			int	*longint*	*longint*
7	-	10		*longint*	*longint*
8	+:=	1	*ref longint*	*longint*	*ref longint*
	-:=	1	*ref longint*	*int*	*ref longint*

9	*	7	*longint* *longint* *int*	*longint* *int* *longint*	*longint* *longint* *longint*
10	*:=	1	*ref longint* *ref longint*	*longint* *int*	*ref longint* *ref longint*
11	**	8	*longint*	*int*	*longint*
12	*over* *mod*	7	*longint* *longint*	*longint* *int*	*longint* *longint*
13	*over ab* *mod ab*	1	*ref longint* *ref longint*	*longint* *int*	*ref longint* *ref longint*
14	*gcd*	2	*longint* *longint* *int*	*longint* *int* *longint*	*longint* *longint* *longint*

1. *copy* *l* generates a copy of *l* ; i.e. a copy of the array of *a* *of* *l* is mode, and delivered in a new structure.

2. *abs* *l* returns the absolute value of *l* .

3. *zero* *l* returns *true* if *l* is *longzero*.

4,...,14. These operators are direct extensions of the operations defined for *int* operands.

C. Formula manipulation

C1 Fundamental declarations

```
mode const   = struct (longint value) ;
mode pol     = struct (string variable,reflits next) ;
mode reflits = ref llterms ;
mode llterms = struct (reflits next, formula coef, int exp) ;
mode formula = union (ref const, ref pol) ;
```

A formula is either a reference to a constant (i.c. a long integer) or a reference to a polynomial in one variable. As the coefficients of the polynomial are formulas by themselves, polynomials in arbitrary many variables can be generated. In order to preserve a unique representation, a lexiographic ordering of the variables is assumed. Thus, a polynomial in x and y , is regarded as a polynomial in x , with polynomials in y as coefficients.
A polynomial is represented by its variable and a list of terms; each term consists of a coefficient and an exponent, which indicates the power of the variable. The terms are ordered such that the greater exponents precede the smaller ones.

```
const  zero  = (const i ; value of i := longzero ; i) ;
const  one   = (const i ; value of i := widen 1 ; i) ;
```

By the declaration of zero and one we obtain the neutral elements for addition and multiplication in the ring of polynomials.

C2 Transformations and selections

	procedure identifier	parameter	result
1	*makenumber*	*longint*	*ref const*
2	*leadingcoeff*	*ref pol*	*formula*
3	*leadvar*	*ref pol*	*string*
4	*degree*	*ref pol*	*int*
5	*algvar*	*string*	*ref pol*

1. *makenumber* (*i*) generates a *consth* on the heap, and *value of h* refers to the *longint* *i* ;
2. *leadingcoeff* (*p*) yields the coefficient of the leading term of *p* ;
3. *leadvar* (*p*) yields the (dominating) variable of *p* ;
4. *degree* (*p*) yields the degree of the polynomial *p* , regarded as polynomial in the dominating variable;
5. *algvar* (*s*) generates the polynomial in *s* with degree 1 and coefficient *one*.

C3 Operators

	operator	prio	left operand	right operand	result
1	= /=	4	*formula*	*formula*	*bool*
2	+ -	6	*formula* *formula* *int*	*formula* *int* *formula*	*formula* *formula* *formula*
3	-	10		*formula*	*formula*
4	*	7	*formula* *formula* *int*	*formula* *int* *formula*	*formula* *formula* *formula*
5	*over*	7	*formula* *formula*	*formula* *int*	*formula* *formula*
6	*gcc*	10 7	 *formula*	*formula* *formula*	*ref const* *ref const*
7	**	8	*formula* *string*	*int* *int*	*formula* *formula*

1. *f* = *g* — returns *true* when *f* and *g* represent the same constant or polynomial;
   f /= *g* — is equivalent to *not* (*f* = *g*) .
2. *f* + *g* (*f* - *g*) — generates a new formula, which is equal to the sum (difference) of *f* and *g*.
3. -*f* — is equivalent to *zero*-*f*.
4. *f* * *g* — generates a new formula, which is equal to the product of *f* and *g*.
5. *f* *over* *g* — generates a new formula, which is equal to the entire quotient of *f* and *g*.

We remark that the operations *f 0 i* and *i 0 f* are equivalent to *f 0 makenumber (widen i)* and *makenumber (widen i) 0 f* ; where *0* stands for one of the operators given above.

6. *gcc f* yields the greatest common divisor of the integer coefficients of the terms of *f* ;

   f gcc g is equivalent in its result to *value of (gcc f) gcd value of (gcc g)* , but probably more efficient.

7. *f ** i* generates a new formula, which is equal to the product $\prod_{k=1}^{i} f$; thus, non positive values of *i* yield *one* as a result.

   *s ** i* is equivalent to *algvar(s) ** i* .

C4 Complexification

string imvar = " i" ;
formula fimvar = algvar(imvar) ;

By means of the formula fimvar operations over the ring of entire complex number are simulated. A complex number is regarded as a polynomial in *imvar*.

	operator	prio	left operand	right operand	result
1	*re*	10		*formula*	*formula*
2	*im*	10		*formula*	*formula*
3	*conj*	10		*formula*	*formula*
4	*mds*	10		*formula*	*formula*
5	*complex*	10		*formula*	*formula*

1. *re f* yields the real part of *f*.
2. *im f* yields the imaginary part of *f*.
3. *conj f* is equivalent to *re f - im f * fimvar*.
4. *mds f* is equivalent to *re f ** 2 + im f ** 2*.
5. *complex f* is equivalent to *re f + im f * fimvar* ; we note, that the result does not necessarily yield a formula equal to *f*, as *f* may contain powers of *imvar* higher than 1.

C5 Substitution

proc substitute = (*strings*, *formula fs*, *f*) *formula* ; (..)

yields as a result a formula equivalent to
$\Sigma\, c_i (fs)^i$ if f equals $\Sigma\, c_i\, s^i$.

D. Procedures for Routh's algorithm

1. *mode polform* = *ref* [ ] *formula* ;

The data structure to represent a row of Routh's array, or the coefficients of a Schur polynomial.

2.

procedure	1st param	2nd param	3rd param	4th param	result
coeftopol	*polform*	*string*	*ref formula*		*void*
poltocoef	*ref formula*	*string*	*polform*		*void*
coeftopoltrans	*polform*	*ref formula*	*ref formula*	*ref formula*	*void*
simplify	*ref polform*				*void*

coeftopol (*c*, *s*, *f*) generates the formula $\sum_i c_i\, s^i$, and assigns it to *f* ;

poltocoef (*f*, *s*, *c*) calculates the coefficients of *f* , regarded as a polynomial in *s*, and assigns these values to the elements of *c* ;

coeftopoltrans (*c*, *f1*, *f2*, *f*) generates the formula $\sum_i c_i\, f1^i f2^{-i}) \; f2$ *upb c* , and assigns it to *f* ;

a combination of *poltocoef* and *coeftopoltrans* may be used in order to perform the transformations (3.11) and (3.12).
simplify (*c*) calculates the greatest common integer divisor of the elements of *c* , and divides all elements by this value.

3.

procedure	parameter	result
routh	*polform*	*polform*
schur	*polform*	*polform*

routh (*c*) calculates the Routh array, according to 3.3*, from the rows

c[n] c[n-2] .....

c[n-1] c[n-3] ..... , n = *upb c* .

Moreover, integer factors occuring in each element of a row, are divided out. As a result, *routh* (*c*) delivers the row $\tilde{r}_0,\ldots,\tilde{r}_n$, possibly dividied by a common factor.

schur (*c*) calculates successively the coefficients of the polynomials P_i ($1 \le i \le n$ = *upb c*), according to (3.9), from the polynomial $P_n = \Sigma\, c_i\, x^i$. As a result, *schur* (*c*) delivers the relations (3.10)

$$|P_i^*(0)|^2 - |P_i(0)|^2 , \quad i = 1,\ldots,n .$$

5. *Examples*

In the analysis of the stability of multistep methods for Volterra integral equations (Van der Houwen, page 14), the polynomial

$$(1) \qquad P_{2n}(x) = \sum_{i=0}^{n} \sum_{j=0}^{n} a_i\{a_j+b_j\,\{(j-i)\,h+c\}\,a\,h\}\,x^{2n-i-j}$$

occurs, which depends on the parameter combinations ah^2 and h a c and the coefficients a_i and b_i . The method will be stable in the (ah^2, hac) plane, if the roots of P_{2n} lie within the unit circle. The Routh's conditions (3.4*) for a class of backward differentiation formulas are obtained by the program:

```
begin
proc  initialize  =  (ref int a,b, int orde) void :
begin  a[0 : orde]:=  case  orde
                      in    (-1,1), (-3,4,-1), (-11,18,-9,2) ,
                            (-25,48,-36,16,-3), (-137,300,-300,200,-75,12) ,
                            (-147,360,-450,400,-225,72,-10) .
                      esac ;
```

```
        b[0]:= case orde in  1,2,6,12,60,60 esac ;
        for i to orde do  b[i] = 0  od
end   #initialize#   ;
proc   derivepol = (ref [ ] int a,b, int n, ref  formula p) void :
begin heap formula   hac := algvar ("hac"), ah2 := algvar("ah2") ,
                       x  := algvar (" x"); #the space is inserted to
                             get the ordering  "  x" < "ah2" < "hac"#
        p := zero ;
        for i from 0 to n do for j from 0 to n do
        p := (a[i] * a[j] + a[i] * b[j] * (hac + ah2 * (j-i)))
                                          * x **(2*n-i-j) + p
        od od
end     #derivpol# ;
proc   f write = (formula f) void : ( ... ) ; # writes f in some nice
                                                format #
for  orde  to 6
do   [0:orde ] int a,b ; initialize (a,b,orde) ;
     formula p, ptrans; derivepol (a,b,orde,p) ;
     fwrite(p) ;
     [0 : 2 * orde ] formula c,c trans, routhcond ;
     poltocoef(p," x",c);
     formula  zp1 := algvar (" z")+one,zm1 := one-algvar("  z") ;
     # transformation from x to (1+z) / (1-z) #
     coeftopoltrans (c,zp1,zm1,ptrans) ; poltocoef (ptrans, " z", ctrans) ;
     simplify (ctrans) ;
     routhcond := routh(ctrans) ;
     for i from 0 to 2 * orde
     do routhcond[i] := routhcond[i] over gcc routhcond[i] ;
          fwrite (routhcond[i])
     od
od
```

We give the results for order = 2; we obtained results for higher orders, too, but these are not suited to be typed here, because of the lengthy formulas.

$$P4(x) = (-6\ hac + 9)\ x^4 + (-8\ ha2 + 8\ hac - 24)\ x^3 + (4\ ah2 - 2hac + 22)x^2 - 8x + 1\ ,$$

$$routh[0] = 3\ ah2 - 4\ hac + 16\ ,$$

$$routh[1] = 2\ ah2 - 5\ hac + 8\ ,$$

$$routh[2] = 2\ ah2^2 + (-11\ hac + 24)\ ah2 + 36\ hac^2 - 68\ hac + 32\ ,$$

$$routh[3] = -4ah2^2 + (-9hac^2 + 8hac)\ ah2 - 9\ hac^3 + 17hac^2 - 8hac\ ,$$

$$routh[4] = -ah2\ .$$

Example 2

The generating polynomials of a fourth order, three step method for first order differential equations are given by

$$\begin{aligned} \rho(x) &= (6a + 2b + 8)\ x^3 + (-6a - 6b)\ x^2 + (-6a + 6b)\ x + 6a - 2b - 8\ , \\ \sigma(x) &= (2a + b + 3)\ x^3 + (6a - b + 9)\ x^2 + (-6a - b + 9)\ x - 2a + b + 3\ . \end{aligned} \tag{2}$$

The method is stable for a value iz on the imaginary axis, if

$$\chi(x) = \rho(x) - iz\ \sigma(x) \tag{3}$$

is a simple Von Neumann polynomial.
The Schur conditions (3.10) are obtained by the following program:

```
begin formula z := algvar(" z") * fimvar, a:= algvar("a"), b := algvar("b") ;
      [0 : 3] formula coef ; [1 : 3] formula schurcond ;
      coef[0] := (6*a-2*b+8) - z * (-2*a+b+3);
      coef[1] := (b-a)*6 - z * (-6*a-b+9);
      coef[2] := (a+b)*-6 - z*(6*a-b+9);
```

```
        coef[3] := (6*a + 2*b +8) - z * (2*a + b  + 3);
        schur cond := schur (coef);
        for i to 3
        do fwrite (schurcond[i]) od
end ;
```

The results are

$$\text{schur}[1] = (-54b^3 - 270b^2 - 378b - 162)\, a^4 z^8 + (-324 b^3 - 1944b^2 - 2916b - 1296)\, a^4 z^6 ,$$

$$\text{schur}[2] = (-3b^2 + 6b + 9)\, a^2 z^4 + (108b + 108)\, a^2 z^2 + (432b + 432)\, a^2 ,$$

$$\text{schur}[3] = (2b + 6)\, a\, z^2 + (12b + 48)\, a .$$

Note, that we did not eliminate common integral factors in this example. By the call fwrite (schurcond[i] over gcc schurcond[i]), we would have obtained more simple expressions (after division by 54,3 and 2).

6. *Source texts*

In this section we give the source texts of the procedures schur and routh, which illustrates the use of the formula manipulation system. The other texts, together with (implementation dependent) procedures for the printing of formulas, are obtainable upon request.

```
proc schur = ( polform p) polform:
  ( int n = upb p; [0:n] formula ploc, heap [1:n] formula cond;
    for i from 0 to n do ploc[i]:=p[i] od;
    for i from n by -1 to 1
    do formula ci:= conj ploc[i]; [0:i-1] formula r;
       for j to i do r[j-1]:= ci*ploc[j] - ploc[0]*conj ploc[i-j] od;
       for j from 0 to i-1 do ploc[j]:= complex r[j] od;
       cond[i]:= ploc[i-1]
    od; cond );

proc routh = ( polform p) polform:
  ( int n = upb p; heap [0:n] formula c;
    int n1= n over 2, n2= (n-1) over 2;
    [0:n1] formula row1, row2; polform first, second, next;
    for i from 0 to n2
    do row2[i]:=p[n-2*i-1];  row1[i]:=p[n-2*i] od;
    if odd n then skip else row2[n1]:=zero; row1[n1]:=p[0] fi;
    first:=row1; second:=row2; simplify(first); simplify(second);
    c[0]:=first[0]; c[n]:=p[0];
    for i from 2 to n-1
    do c[i-1]:=second[0]; int i2=(n-i) over 2;
       next:= if odd i then row2 else row1 fi;
       for j from 0 to i2
       do next[j]:= c[i-1]*first[j+1] - c[i-2]*second[j+1];
          if i>3 then next[j]:=next[j] over c[i-3] fi
       od;
       next[i2+1]:= zero; simplify(next);
       first:= second; second:= next
    od; if n>1 then c[n-1]:= second[0] fi;
    c);
```

Bibliography

Barreiss, E.H. *Sylvester's identity and multistep integer-preserving Gaussian elimination*, Math. Comp. 22, 565-578 (1968).

Barnett, S and D.D. Siljak, *Routh's algorithm: a centennial survey*, SIAM Review, volume 19, 472-489 (1977).

Houwen, P.J. van der and P.H.M. Wolkenfelt, *On the stability of multistep formulas for Volterra integral equations of the second kind*, Report NW 59, Mathematisch Centrum, Amsterdam (1978).

Knuth, D.E., *The art of computer programming, Vol. I : Fundamental algorithms*, Addison-Wesley, Reading, Massachusetts (1968).

Lambert, J.D., *Computational methods in ordinary differential equations*, John Wiley & Sons, London (1973).

Marden, M., *Geometry of polynomials*, Amer. Math. Soc., Providence (1966).

Meulen, S.G. van der and M. Veldhorst, TORRIX, *a programming system for operations on vectors and matrices over arbitrary fields and of variable size*, volume 1, MC Tracts 86, Mathematisch Centrum, Amsterdam (1978).

Miller, J.J.H., *On the location of zeros of certain classes of polynomials with applications to numerical analysis*, J. Inst. Math. Appl., 8, 397-406 (1971).

Teer, F., *A polynomial manipulation program in ALGOL 68*, IR-28, Vrije Universiteit, Amsterdam (1978).

Wijngaarden, A. van et al. (Eds), *Revised report on the algorithmic language ALGOL 68*, MC Tracts 50, Mathematisch Centrum, Amsterdam (1976).

Wirth, N., *Systematisches Programmieren, Teubner* (1972).

Stability of linear multistep methods on the imaginary axis

by

K. Dekker

ABSTRACT

The stability of linear multistep methods of order higher than one is investigated for hyperbolic equations. By means of the Routh array and the Hermite-Biehler theorem, the stability boundary on the imaginary axis is expressed in terms of the error constant of the third order term. As a corollary we state the result that the stability boundary for methods of order higher than two, is at most $\sqrt{3}$, and this value is attained by the Milne-Simpson method.

KEY WORDS & PHRASES: Numerical analysis, Linear multistep methods, Hyperbolic equations, Stability analysis.

1. INTRODUCTION

For the initial value problem

$$y' = f(x,y), \quad y(0) = y_0, \tag{1.1}$$

the linear k-step method is defined by

$$\sum_{j=0}^{k} \alpha_j y_{n+j} - h \sum_{j=0}^{k} \beta_j f(x_{n+j},y_{n+j}), \quad n=0,1,\ldots . \tag{1.2}$$

In this paper we will study the behaviour of the difference equation (1.2) on *hyperbolic* problems; thus, the Jacobian of (1.1)

$$\frac{\partial f}{\partial y} \tag{1.3}$$

has purely imaginary eigenvalues. Application of (1.2) to the model equation

$$y' = \lambda y, \quad y(0) = 1, \tag{1.4}$$

leads to

$$(\rho(E) - h\lambda \sigma(E)) y_n = 0, \quad n=0,1,\ldots , \tag{1.5}$$

where E denotes the shift operator $Ey_n = y_{n+1}$, and ρ and σ are the polynomials

$$\rho(\xi) = \sum_{j=0}^{k} \alpha_j \xi^j, \quad \sigma(\xi) = \sum_{j=0}^{k} \beta_j \xi^j. \tag{1.6}$$

It is well known that all solutions of (1.5) are bounded if and only if $q = h\lambda$ lies in the stability region S, defined by

$$S = \{ q \in \mathbb{C} \mid \rho(\xi) - q\sigma(\xi) = 0 \Rightarrow (|\xi| < 1 \text{ or } |\xi| = 1 \text{ and } \xi \text{ is a simple root}) \} . \tag{1.7}$$

DEFINITION: A linear multistep method is said to be *stable* on the imaginary axis if $\{ iy \mid -\infty < y < \infty \} \subset S$.

DEFINITION: The *imaginary stability boundary* of a multistep method is the largest number w_0, such that $\{ iw \mid -w_0 < w < w_0 \} \subset S$. In the remainder of this paper we will call w_0 briefly the stability boundary.

JELTSCH[5] has proved that the highest order for a consistent linear multistep method which is stable on the imaginary axis, is two; in his proof the well-known theorem of DAHLQUIST[3] about *A-stability* and order of a multistep method is used.

This result is entirely different from those obtained for *parabolic* equations (i.e. the eigenvalues of (1.3) are negative). For instance, CRYER [2] showed that there are linear multistep methods of arbitrarily high order, which are stable along the negative real axis. Of late we have been trying to construct linear multistep methods of order at least three with an optimal stability interval along the imaginary axis. To that end we implemented Routh's algorithm (see BARNETT & SILYAK [1]), using a formula manipulation program (DEKKER [4]) and tried to optimize the stability boundary. Despite many efforts we were not able to exceed $\sqrt{3}$, the stability boundary of the Milne-Simpson method, which has order four. In this paper we proof that the stability boundary of any linear multistep method of order higher than two, is really at most $\sqrt{3}$.

During our investigations, we received a personal communication from Jeltsch, stating the same result. His proofs, based on the algebraic techniques described in JELTSCH & NEVANLINNA [6], will appear in the near future in a joint paper of these authors.

2. CONSISTENCY CONDITIONS

In the analysis of multistep methods it is convenient to map the unit circle $|\xi| < 1$ onto the left half plane $\mathrm{Re}(z) < 0$, by the transformations (see CRYER [2], VARAH [10]),

$$(2.1) \qquad z = \frac{\xi + 1}{\xi - 1}, \quad \xi = \frac{z + 1}{z - 1}.$$

The polynomials $\rho(\xi)$ and $\sigma(\xi)$ are transformed into

$$(2.2)\qquad \begin{aligned} r(z) &= 2^{-k}\,(z-1)^k \rho\left(\frac{z+1}{z-1}\right) = \sum_{j=0}^{k} a_j\, z^j, \\ s(z) &= 2^{-k}\,(z-1)^k \sigma\left(\frac{z+1}{z-1}\right) = \sum_{j=0}^{k} b_j\, z^j. \end{aligned}$$

The stability region S may be defined in terms of the new polynomials:

$$(2.3)\qquad S = \{q\in\mathbb{C} \mid r(z) - q\,s(z) = 0 \Rightarrow (\mathrm{Re}(z)<0 \text{ or } \mathrm{Re}(z)=0 \text{ and } z \text{ is a simple root})\}\ ,$$

which is equivalent to (1.7).

The error constants of a method are usually defined by formulas, linear in the coefficients α_j and β_j (see LAMBERT [7], page 23). For convenience, we introduce the modified error constants $\tilde{C}_j$, defined by

$$(2.4)\qquad \tilde{C}_j = a_{k-j} - 2 \sum_{m=0} \frac{b_{k-j+1+2m}}{1+2m}\ , \quad j=0,1,\ldots\ ;$$

these constants differ a factor from the constants given by Lambert. A method is said to be *of order p*, if $\tilde{C}_0,\ldots,\tilde{C}_p$ are equal to zero, and if $\tilde{C}_{p+1} \neq 0$ (cf. CRYER [2]).

REMARK: In equation (2.4) and throughout the remainder of this paper, we omit the upper index of the summation; we intend this to be the largest value, for which the term is non-zero. Moreover, we assume $a_j=b_j=0$ if $j<0$ or $j>k$.
Obviously, for a consistent method a_k equals zero; the scaling factor b_k is chosen equal to 1.

3. STABILITY

In order to determine the stability of a multistep method, we have to locate the zeros of a polynomial in z of degree k

$$(3.1)\qquad f(z,q) = r(z) - q\,s(z).$$

To facilitate the notations, we introduce the following sets in the complex plane:

$$H_- = \{ z \in \mathbb{C} \mid \mathrm{Re}(z) < 0 \},$$
$$H_+ = \{ z \in \mathbb{C} \mid \mathrm{Re}(z) > 0 \},$$
$$R_- = \{ z \in \mathbb{R} \mid z < 0 \},$$
$$R_+ = \{ z \in \mathbb{R} \mid z > 0 \},$$
$$I = \{ z \in \mathbb{C} \mid \mathrm{Re}(z) = 0 \},$$

and we denote their closures by $\overline{H}_-$, $\overline{H}_+$, etc. .

DEFINITION: We call a polynomial *stable* if all its roots lie in $\overline{H}_-$.

Obviously, whenever q lies in S, defined by (2.3), then the polynomial f(z,q) is stable.

Throughout this paper we will assume that r(z) is stable, i.e. that $0 \in S$, that f(z,q) is non-reducible, i.e. r and s have no common roots, and that q is purely imaginary.

The *Routh array* forms a useful tool, to determine whether the zeros of a polynomial lie in H_-, H_+, R_-, R_+, $\overline{H}_-$, etc. . Theorems about the application of the Routh array may be found in MARDEN [8, Chapters 9 and 10] ; BARNETT & SILYAK [1] give a useful survey. According to BARNETT & SILYAK [1, section 3.5] the number of roots in H_- of a complex polynomial, given by

$$f(z) = \alpha_k z^k + (\alpha_{k-1} + i\,\alpha'_{k-1})\, z^{k-1} + \ldots + (\alpha_0 + i\,\alpha'_0) \tag{3.2}$$

may be found by forming Routh's array, with initial rows

$$\begin{array}{cccccc} \alpha_k & \alpha'_{k-1} & -\alpha_{k-2} & -\alpha'_{k-3} & \cdots \\ \alpha_{k-1} & \alpha'_{k-2} & -\alpha_{k-3} & -\alpha'_{k-4} & \cdots . \end{array} \tag{3.3}$$

For a regular array, the number of roots in H_+ equals the number of variations in sign in the sequence formed by the first elements of these rows. However, the array is not regular for a multistep method of order $p \geq 2$, as the first element of the third row, defined by

$$\alpha'_{k-1} - \alpha'_{k-2} \frac{\alpha_k}{\alpha_{k-1}}$$

turns out to be zero. Thus we proceed in a slightly different way. The rows (3.3) may be regarded as a representation of two real polynomials

$$\begin{aligned} f_0(y) &= \alpha_k y^k + \alpha'_{k-1} y^{k-1} - \alpha_{k-2} y^{k-2} - \alpha'_{k-3} y^{k-3} + \ldots, \\ f_1(y) &= \alpha_{k-1} y^{k-1} + \alpha'_{k-2} y^{k-2} - \alpha_{k-3} y^{k-3} - \alpha'_{k-4} y^{k-4} + \ldots . \end{aligned} \tag{3.4}$$

The correspondence between the real variable y and the imaginary variable z, y=-iz, will be assumed throughout the rest of this paper.

It is obvious, that the roots of f(z) are purely imaginary, if the roots of $f_0(y)$ are real, and $f_1(y)$ is identically equal to zero. Now we will proof that the stability of f(z) implies that all roots of $f_0(y)$ are real, whether or not $f_1(y)$ is the zero-function. At first, we modify the *Hermite-Biehler theorem* (see OBRESCHKOFF [9, page 106] or MARDEN [8, page 169]).

THEOREM 3.1: *(Hermite-Biehler) All roots of the polynomial* f(y)=u(y)+iv(y), *where* u *and* v *are real polynomials, lie on the same side of the real axis, if and only if* u *and* v *have simple real roots which separate each other.*

As we need a result about the left half-plane H_-, we have to rotate the complex plane.

COROLLARY 3.1: *All roots of the polynomial* f(z), *such that* f(iy)=u(y)+iv(y). *lie on the same side of the imaginary axis, if and only if the real polynomials* u(y) *and* v(y) *have simple real roots which separate each other.*

PROOF: z=iy is a root of f, if and only if y is a root of u+iv. □

In the following lemma we include roots lying on the imaginary axis.

LEMMA 3.2: *Let* f *be a complex polynomial, such that* f(z)=f(iy)=u(y)+iv(y), *where* u *and* v *are real polynomials. If all roots of* f *lie in* $\overline{H}_-$, *then all roots of* u *and* v *are real.*

PROOF: Assume that f(z) has m zeros, $z_1,\ldots,z_m$, on the imaginary axis. Obviously, u(y) takes real values and iv(y) purely imaginary values, if $y\in\mathbb{R}$. Thus, the real points $-iz_j$, $j=1,\ldots,m$, are zeros of both u and v.
Now consider the polynomials $\tilde{f}$, $\tilde{u}$ and $\tilde{v}$, which are obtained from f, u and v by dividing these polynomials by the common factors of u and v. The zeros of $\tilde{f}$ are the remaining zeros of f, and lie in H_-. Moreover, $\tilde{u}$ and $\tilde{v}$ are real polynomials, and the relation $\tilde{f}(z)=\tilde{f}(iy)=\tilde{u}(y)+i\tilde{v}(y)$ holds. Thus, according to COROLLARY 3.1, the roots of $\tilde{u}$ and $\tilde{v}$ are real and simple, and separate each other. We conclude that all roots of u and v are simple. □

REMARK 3.1: The roots of u (or v) need not be simple, even if all roots of f are simple; the roots of $\tilde{u}$ (or $\tilde{v}$) may coincide with those produce by the purely imaginary roots of f. For example, $f(z)=(z+1)(z-i)(1-i)$ has simple roots in $\bar{H}_-$, but u(y), obtained from $f(iy)=-(y^2-2y+1)+i(y^2-1)$, has a double root.

COROLLARY 3.2: *A necessary condition for the stability of* f(z), *given by (3.2), is that all roots of the polynomials* $f_0(y)$ *and* $f_1(y)$, *as defined by (3.4), are real.*

PROOF: It is easily verified that f(iy) equals $i^k f_0(y) + i^{k-1} f_1(y)$, and that both f_0 and f_1 are real polynomials. The stability of f implies that all roots of f lie in $\bar{H}_-$, and thus all roots of f_0 and f_1 are real. □

Now, we apply these results to the polynomial f(z,q), defined by (3.1). Using the expressions (2.2) for r(z) and s(z), and multiplying with i to make the first coefficient real (note that $a_k=0$), we get

$$(3.4')\qquad \begin{aligned} f_0(y,w) &= \sum_{j=0} (a_{k-1-2j} + b_{k-2j}\, w\, y)\,(-1)^j\, y^{k-1-2j}, \\ f_1(y,w) &= \sum_{j=0} (a_{k-2-2j} + b_{k-1-2j}\, w\, y)\,(-1)^j\, y^{k-2-2j}, \end{aligned}$$

where we made the substitution w=-iq to shorten the notation. Thus, according to COROLLARY 3.2, a necessary condition for the stability of f(z,q) is that all roots of $f_0(y,w)$ and $f_1(y,w)$ are real.

EXAMPLE 3.1: The two-step Curtiss-Hirschfelder formula yields the polynomial

$$f(z,q) = 2\,z + 4 - q\,(z^2 + 2\,z + 1).$$

The initial rows of the Routh array are

$$\begin{array}{lll} -iq & 2 & iq \\ -2iq & 4 & \end{array}$$

and the polynomials f_0 and f_1, according to (3.4')

$$f_0(y,w) = w\,y^2 + 2\,y - w,$$
$$f_1(y,w) = 2w\,y + 4.$$

Both polynomials have real zeros for real values of w, so the condition of COROLLARY 3.2 is satisfied. Moreover, the zeros separate each other, which implies, according to COROLLARY 3.1, that all zeros of f(z,q) lie in the same half plane. As the roots are continuous functions of q, and the root of f(z,0) lies in H_-, we conclude that all roots of f(z,q) lie in H_-, for all $q \in I$. We note, that we did not state this stronger result about the separation of the roots in LEMMA 3.2, because we disregard the polynomial f_1 in the sequel.

EXAMPLE 3.2: The three-step Curtiss-Hirschfelder formula yields the polynomials

$$\rho(\xi) = \frac{4}{3}\,(11\xi^3 - 18\xi^2 + 9\xi - 2), \qquad \sigma(\xi) = 8\xi^3,$$
$$r(z) = 2z^2 + 6z + \frac{20}{3}, \qquad s(z) = z^3 + 3z^2 + 3z + 1,$$
$$f_0(y,w) = w(y^3 - 3y) + 2y^2 - \frac{20}{3}, \qquad f_1(y,w) = w(3y^2 - 1) + 6y.$$

$f_0(y,w)$ has three real roots if $|w| < \frac{1}{3}\sqrt{5}$ or $|w| > \frac{1}{3}\sqrt{32}$; $f_1(y,w)$ has real roots for all values of w. The condition for the roots to separate each other are found by the Routh array, deleting the leading zeros. We get $16w^2 - 60 > 0$, so the formula is unstable on $\{\, iw \mid -\frac{1}{2}\sqrt{15} < w < \frac{1}{2}\sqrt{15} \,\}$.

Observing that the error constants $\tilde{C}_j$, defined by (2.4), contain coefficients of f_0 if j is odd, and coefficients of f_1 if j is even, leads to

THEOREM 3.3: *Suppose there exists a k-step formula of order p (p odd) with stability boundary* w_0. *Then there exists also a k-step formula of order at least p+1, whose associated polynomial* $\tilde{f}(z,q)$ *is stable if* $-w_0 < iq < w_0$.

PROOF: Let the associated polynomial (3.1) of the k-step formula of order p be given by f(z,q). According to COROLLARY 3.2, all roots of the polynomial $f_0(y,w)$ are real, if $|w|<w_0$. Now, choose $\tilde{f}(z,q)$ in such a way, that the polynomials $\tilde{f}_0$ and $\tilde{f}_1$, generated by $\tilde{f}$ according to (3.4'), are equal to f_0 and the zero-function, respectivily. Thus

$$\tilde{f}(z,q) = i^{k-1} f_0(-iz,-iq) = i^{k-1} f_0(y,w).$$

If $|q|<w_0$, then all roots of f_0 are real, and all roots of $\tilde{f}$ purely imaginary; thus $\tilde{f}(z,q)$ is stable. Moreover, the error constants $\tilde{C}_j$ are equal to zero, if $j \leq p$ or if j is even, so the order of the new formula is at least p+1. □

EXAMPLE 3.3: For the Backward Euler formula we have

$$f(z,q) = 2 - q\,(z + 1);$$

thus, $f_0(y,w) = wy + 2$ and $f_1(y,w) = w$. f(z,q) is stable for imaginary values of q, and we have first order consistency, as is easily checked by using (2.4). Now, we choose $\tilde{f}_0(y,w) = wy + 2$ and $\tilde{f}_1(y,w) = 0$, which yields $\tilde{f}(z,q) = 2 - q\,z$. The resulting formula is the trapezoidal rule, which is known to be stable on I and which is of second order.

When we have a multistep method of second order, we may have stability on the whole imaginary axis. Now, we will investigate what happens if we increase the order. In that case the leading terms of $f_0(y,w)$ are

$$w\,y^k + 2\,y^{k-1} - w\,b_{k-2}\,y^{k-2} - (2b_{k-2} + \tfrac{2}{3})\,y^{k-3} + \ldots \quad .$$

As the stability interval $\{ iw \mid -w_0 < w < w_0 \}$ is symmetric around the

origin, we should consider $f_0(y,w)$ for both positive and negative values of w. It is therefore convenient to form the product of $f_0(y,w)$ and $f_0(y,-w)$; this polynomial is quadratic in y. Depending on the variable, y or y^2, we will denote this polynomial by $g(y,w)$ and $h(x,w)$, respectivily. Finally, separating terms containing the factor w from the other ones, we obtain the following polynomials:

$$
\begin{aligned}
g_0(y) &= \{ \sum_{j=0} (-1)^j \, b_{k-2j} \, y^{k-2j} \}^2, \\
g_1(y) &= \{ \sum_{j=0} (-1)^j \, a_{k-1-2j} \, y^{k-1-2j} \}^2, \\
h_0(x) &= h_0(y^2) = g_0(y), \\
h_1(x) &= h_1(y^2) = g_1(y).
\end{aligned}
\tag{3.5}
$$

It is easily verified that g_0 and g_1 satisfy the relation

$$
g_1(y) - w^2 \, g_0(y) = f_0(y,w) \; f_0(y,-w), \tag{3.6}
$$

so that $g(y,w)$ equals $g_1(y) - w^2 g_0(y)$.

LEMMA 3.4: *A necessary condition for the stability of* $r(z)$ $(=f(z,0))$ *is that all roots of* h_1 *are non-negative.*

PROOF: The stability of $f(z,0)$ implies that all roots of $f_0(y,0)$ are real; however, $f_0(y,0)$ is an odd (or even) function, so $-y_0$ is a root of $f_0(y,0)$ if y_0 is a root of $f_0(y,0)$. Assume that $y_1,\ldots,y_{k-1}$ are the real roots of $f_0(y,0)$; then $-y_1,\ldots,-y_{k-1}$ are also roots of $f_0(y,0)$. Hence, using (3.6), we see that the factors of $g_1(y)$ are $(y-y_j)(y+y_j)$, $j=1,\ldots,k-1$, and the factors of $h_1(x)$ are $(x-y_j^2)$. We may conclude that the roots of h_1 are non-negative. □

LEMMA 3.5: *Assume that* g_0 *and* g_1 *do not have common roots. A necessary condition for the stability of* $f(z,q)$, *for all* $q \in I$, $|q|<\varepsilon$, *for some small* $\varepsilon>0$, *is that all roots of* g_1 *are double.*

PROOF: Let y_0 be a zero of g_1 of order at least 4. By assumption, y_0 is not a zero of g_0. As g_0 is a quadratic function, we have $g_0(y_0)>0$, so that there are only two real zeros of $g(y,w)$ in the neighbourhood of y_0, if w^2 is small enough. As the roots of $g(y,w)$ are continuous functions of w, we must have two complex(non-real) zeros of g; thus at least one of the functions $f_0(y,w)$ and $f_0(y,-w)$ has a non-real zero, which, according to COROLLARY 3.2, would imply that $f(z,q)$ is not stable, if $q=iw$ or $q=-iw$. As a consequence, the stability of $f(z,q)$ for all $q, |q|<\varepsilon$, implies that the zeros of g_1 are of order less than 4. Moreover, by definition (3.5), the roots of g_1 are of even order, so they are of order 2. □

REMARK 3.2: We can not replace g_1 by h_1 in this lemma, because 0 can be a single root of h_1. The non-zero roots of h_1, however, are always double, so it depends on the degree of the polynomial h_1, whether 0 is a (single) root or not. According to (3.5), the degree of h_1 is k-1.

LEMMA 3.6: *A necessary condition for the stability of the polynomials* $f(z,q)$ *and* $f(z,-q)$, *for a fixed* $q\in I$, *is that all roots of* $g(y,w)$, $w=-iq$, *are real.*

PROOF: COROLLARY 3.2 states that the stability of $f(z,q)$ implies that all roots of $f_0(y,w)$ are real. Likewise, all roots of $f_0(y,-w)$ are real as a consequence of the stability of $f(z,-q)$. Thus, according to relation (3.6), all roots of $g(y,w)$ are real. □

In order to get an idea of the behaviour of the zeros of g for various values of w, it is convenient to consider the function

$$(3.7) \qquad Q(y) = \frac{g_1(y)}{g_0(y)} = w^2 + \frac{g(y,w)}{g_0(y)}.$$

It is clear that all zeros of $g(y,w)$ are real, if and only if the function $Q(y)$ has $2k$ points in common with the constant function w^2. In figure 3.1 we have plotted the function $\{Q(y)\}^{\frac{1}{2}}$ for the three-step Curtiss-Hirschfelder formula (see EXAMPLE 3.2).

When we look at the plot of figure 3.1, we get an idea of the intervals of stability and instability of the three-step Curtiss-Hirschfelder formula.

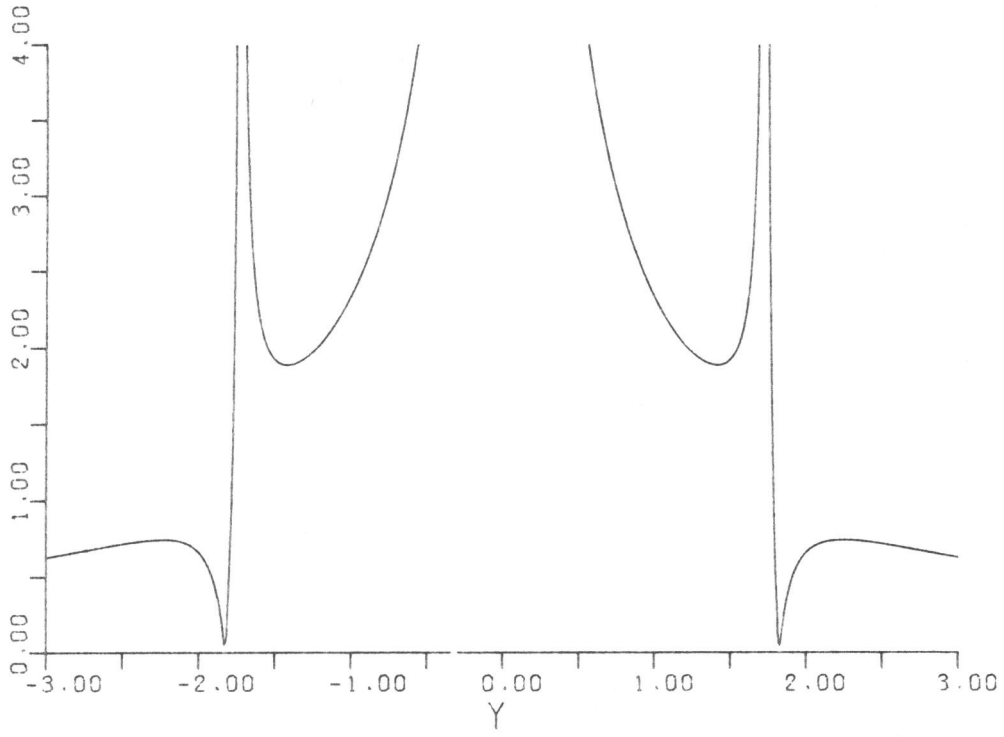


Fig. 3.1 The function $(Q(y))^{\frac{1}{2}} = 2\left(\frac{(y^2 - 10/3)^2}{y^2 (y^2 - 3)^2}\right)^{\frac{1}{2}}$.

$Q(y)$ has two double zeros, and $\lim_{y\to-\infty} Q(y) = \lim_{y\to\infty} Q(y) = 0$; this indicates that $g(y,0)$ has two double real zeros, whereas the degree is 4 (=2k-2). Thus, all roots of $g(y,0)$ are real. For small values of w ($|w|<\sqrt{5}/3$), there are 6 points of intersection between $Q(y)$ and the constant function w^2; hence, all roots of $g(y,w)$ are real for these values of w. We remark that each of the intervals $(-\infty,-\sqrt{30}/3)$, $(-\sqrt{30}/3,+\sqrt{30}/3)$ and $(\sqrt{30}/3,\infty)$ contains exactly two intersection points. For values of $w>\sqrt{5}/3$, these intersection points vanish in two of those intervals. These intersection points can not "jump" at once into the other interval (otherwise $g(y,\sqrt{5}/3)$ would have had 4 double and two single roots), so $\sqrt{5}/3$ is an upperbound for the stability boundary of the formula

In the remainder of this section we will prove that the stability boundary can be bounded by the top of the lowest "hill" of the function $\sqrt{Q(y)}$, and we will calculate an upperbound for this top. At first we give four lemma's for rather general polynomials satisfying some conditions. These lemma's can be applied to the real polynomials $h_0(x)$ and $h_1(x)$ as defined in (3.5), and the reader may keep them in mind. We note that h_1 has

$(k-1)\div 2$ double roots, if the conditions of LEMMA 3.5 are satisfied. We denote them by $\tilde{c}_2,\ldots,\tilde{c}_m$, where m equals $(k+1)\div 2$.

Let $c_1,\ldots,c_m$ be real constants, ordered in such a way that $c_1<c_2<\ldots<c_m$. Then we define the open intervals I_j, $j=1,\ldots,m$, by

$$\begin{aligned} I_j &= (\, c_j,\, c_{j+1}), \quad j=1,\ldots,m-1, \\ I_m &= (\, c_m,\, \infty\,). \end{aligned} \tag{3.8}$$

In the following four lemma's we assume these constants and intervals be given.

LEMMA 3.7: *Let* $h(x)$ *be a real polynomial of degree* m,

$$h(x) = x^m + \sum_{j=1}^{m}(-c_j)\, x^{m-1} + h_{m-2}x^{m-2} + \ldots + h_0. \tag{3.9}$$

such that c_j, $j=1,\ldots,m$, *is not a zero of* $h(x)$.
Define the polynomial $P(x)$ *by*

$$P(x) = \prod_{j=1}^{m}(x - c_j)^2 - \{h(x)\}^2. \tag{3.10}$$

Then there exists an interval I_j, $1\le j\le m$, *such that* $P(x)<0$, $\forall x \in I_j$.

PROOF: $P(x)$ is a polynomial of degree less than or equal to 2m-2; thus, P has at most 2m-2 roots. As $P(c_j)<0$, for all j, the number of zeros in each of the intervals I_j, $j<m$, is even. Thus, in at least one interval P has no roots and is consequently strictly negative. □

REMARK 3.3: The assumptions of this lemma can be satisfied only, if $m>1$. If $m=1$, we have $h(x) = x - c_1$, which contradicts the assumption that c_1 is not a root of $h(x)$.

REMARK 3.4: In the interval I_j indicated in this lemma, the function $\{h(x)\}^2$ is obviously positive; hence, the function $1 + P(x)/\{h(x)\}^2$ has at least one hill with a top less than 1. If we choose $c_2,\ldots,c_m$ equal to the double roots of $h_1(x)$, $h(x)$ equal to $\sqrt{h_0(x)}$ if $k=2m$, else equal to $\sqrt{x\, h_0(x)}$,

and c_1 such that the second coefficient of $h(x)$ equals $\sum_{j=1}^{m}(-c_j)$, then $Q(y)$ and $1 + P(y^2)/\{h(y^2)\}^2$ differ a factor $(y^2-c_1)^2/(4y^2)$, as the reader may verify easily. This factor is bounded by $|c_1|$ for $y \in \mathbb{R}$, if c_1 is negative. In that case, $Q(y)$ can be bounded by $|c_1|^{-1}(1 + P(y^2)/\{h(y^2)\}^2)$.

LEMMA 3.8: *Let* $h(x)$ *be a real polynomial of degree* m, *with leading terms given by (3.9). Let* c_1 *be negative, and* $c_2,\ldots,c_m$ *be positive. Define* $R(x,w)$ *by*

$$R(x,w) = 4x \prod_{j=2}^{m}(x - c_j)^2 - w^2\{h(x)\}^2. \tag{3.11}$$

Define $\tilde{w}_0 = |c_1|^{-\frac{1}{2}}$. *Then there exists an interval* I_j, $1\le j\le m$, *such that* $R(x,w)$ *does not have roots in* I_j, *if* $|w|>\tilde{w}_0$. *Moreover, if* $h(c_{j'})\neq 0$, *for some* j', $2\le j'\le m$, *then also* $R(x,\tilde{w}_0)$ *does not have roots in the interval* I_j.

PROOF: For all x the relation $4x \le \tilde{w}_0^{\,2}(x-c_1)^2$ holds; equality occurs for $x=-c_1$. Using this relation, we get the inequalities, assuming $|w| \ge \tilde{w}_0$,

$$\frac{R(x,w)}{\tilde{w}_0^{\,2}} \le \prod_{j=1}^{m}(x-c_j)^2 - \frac{w^2}{\tilde{w}_0^{\,2}}\{h(x)\}^2 \le \prod_{j=1}^{m}(x-c_j)^2 - \{h(x)\}^2.$$

If $h(c_j)\neq 0$, for all j, then we apply the previous theorem, and conclude that $R(x,w)$ does not have zeros in I_j, for some j, if $|w| \ge \tilde{w}_0$.
If $h(c_j)=0$, for some j, we may cancel the factors $(x-c_j)^2$, and arrive at the same result, as is easily verified.
If $h(c_j)=0$, for all j, we obtain after division of R by $\prod_{j=2}^{m}(x-c_j)^2$ the function $4x - w^2(x-c_1)^2$; obviously, there is no zero if $|w|>\tilde{w}_0$, and $-c_1$ is a zero if $|w|=\tilde{w}_0$. Thus, there are no roots of $R(x,w)$ in I_1, if $|w| > \tilde{w}_0$. □

EXAMPLE 3.4: The Milne-Simpson method yields the polynomials

$$\rho(\xi) = 2(\xi^2-1), \qquad \sigma(\xi) = \frac{2}{3}(\xi^2+4\xi+1),$$
$$r(z) = 2z, \qquad s(z) = z^2 - 1/3,$$
$$f_0(y,w) = w y^2 + 2y + w/3, \qquad f_1(y,w) = 0,$$
$$g(y,w) = 4y^2 - w^2(y^2+1/3)^2,$$
$$h_0(x) = (x + 1/3)^2, \qquad h_1(x) = 4x.$$

Choose $c_1=-1/3$ and $h(x)=(x+1/3)$. Then the conditions of LEMMA 3.8 are satisfied; thus $4x - w^2(x+1/3)^2$ does not have a real zero, if $|w|>\tilde{w}_0=\sqrt{3}$, and likewise $g(y,w)$ does not have real zeros for these values of w. If $|w|<\tilde{w}_0$, then all roots of $g(y,w)$, and also of $f_0(y,w)$ are real, and they are simple if $|w|\neq\tilde{w}_0$. Thus, the Milne-Simpson method is stable for $q\in I$, if $|q|<\sqrt{3}$.

EXAMPLE 3.5: For the trapezoidal rule we have (see also EXAMPLE 3.3)

$$f_0(y,w) = wy + 2,$$
$$g(y,w) = 4 - w^2 y^2,$$

and we may set

$$R(x,w) = x\, g(\sqrt{x},w) = 4x - w^2 x^2.$$

We have to choose $c_1=0$, in order to satisfy (3.9); however, this value is not negative, and we can not apply LEMMA 3.8. Obviously, $g(y,w)$ has two real zeros for all real values of w.

REMARK 3.5: If the conditions of LEMMA 3.8 are satisfied, then the function $w^2 + R(x,w)/\{h(x)\}^2$ has at least one hill, and the top of this hill is at most equal to $\tilde{w}_0^2$. We note that the actual top can be smaller, due to the (possible not sharp) inequalities used in the proof of this lemma.

LEMMA 3.9: *Let* $R(x,w)$ *be a polynomial in* x *of degree* $2m$ *for* $w\neq 0$ *and of degree at most* $2m$ *for* $w=0$ *and let the coefficients of* R *be real continuous functions of* w. *Assume*

(i) $R(x,0)$ *has* $m-1$ *double real zeros,* $c_2,\ldots,c_m$;

(ii) $R(x,w)$ *does not have zeros in* $c_1,\ldots,c_m$, *if* $w\neq 0$;

(iii) $R(x,w)$ *has two zeros in each of the intervals* $I_1,\ldots,I_m$ *if* $0<|w|\leq\varepsilon$;

(iv) $\exists j$, $1\leq j\leq m$, *such that* $R(x,\tilde{w}_0)$ *has no zeros in* I_j.

Then there exists a w_1, $0<w_1\leq\tilde{w}_0$, *such that* $R(x,w_1)$ *has at most* $2m-2$ *zeros.*

PROOF: Let $n_j(w)$ denote the number of roots of $R(x,w)$ in the interval I_j, $1\leq j\leq m$, double roots counting double. Define the sets S_H and S_V by

$$S_H = \{\, w>0 \mid \forall j,\ 0<\tilde{w}\leq w,\ n_j(\tilde{w})\geq 2 \,\},$$

$$S_V = \{\ w>0 \mid \exists\ j,\ \exists\ \tilde{w}<w,\ n_j(\tilde{w}) < n_j(w)\ \}.$$

We note that both sets are closed, as the roots can not move across the boundaries of the intervals ($R(c_j,w)\neq 0$ if $w>0$, $\forall j$) and they can not vanish (the degree is $2m$ for $w>0$). Moreover, the first set is not empty, because $\varepsilon \in S_H$, and is bounded because $\tilde{w}_0 \notin S_H$. We define the real numbers $w_H = \max\{\ x \mid x\in S_H\}$ and $w_V = \min\{\ x \mid x\in S_V\}$, if S_V is not empty, and $w_V = 2w_H$ if S_V is empty. We may think of w_H and w_V as the top of the hill and the bottom of the valley of a function like $Q(y)$ in figure 3.1.

From $\sum_{j=1}^{m} n_j(w) = 2m$, $0 < w \leq w_H$, we conclude that $w_H < w_V$; otherwise, there would be a w, $w < w_V \leq w_H$ with $\sum_{j=1}^{m} n_j(w) > 2m$. Thus, the total number of roots in the intervals, $\sum_{j=1}^{m} n_j(w)$ is less than $2m$ for $w_H < w < w_V$. Moreover, none of the points c_j is a root of $R(x,w)$, and the total number of real roots must be even, so we arrive at the assertion of the lemma. □

LEMMA 3.10: *Let $c_2,\ldots,c_m$ be positive constants, $h(x)$ a real polynomial satisfying (3.9) and c_1 be negative. Then, for all $\varepsilon>0$, there exists a w_1, $0 < w_1 < (-c_1)^{-\frac{1}{2}} + \varepsilon$, such that the polynomial $R(x,w_1)$, as defined by (3.]]) has at most 2m-2 real zeros.*

PROOF: We may assume without loss of generality that $h(x)$ does not have roots in common with $\prod_{j=2}^{m}(x-c_j)^2$. If there are common roots, we divide both polynomials by the common factors, and apply the proof to the resulting polynomials. We verify that R satisfies the assumptions of LEMMA 3.9:

(i) The real constants c_2, ..., c_m are double roots of $R(x,0)$;

(ii) $R(c_j,w)\neq 0$, $j=2,\ldots,m$, if $w\neq 0$ and $R(c_1,w) < 0$, as c_1 is negative;

(iii) For small values of w, $R(x,w)$ has $2m-2$ zeros in the neighbourhood of the points c_j, $j=2,\ldots,m$, one on each side of each point, and two zeros in the neighbourhood of the points $x=0$ and $x=\infty$. Thus, $R(x,w)$ has two zeros in I_j, $j=1,\ldots,m$, if $0 < |w| < \varepsilon$. (In I_1 because $c_1 < 0 < c_2$)

(iv) According to LEMMA 3.8, $R(x,w)$ has no zeros in I_j for some j, if $|w| > (-c_1)^{-\frac{1}{2}}$. Thus, for all $\varepsilon>0$, $R(x,w)$ has no zeros in I_j, if $w = (-c_1)^{-\frac{1}{2}} + \varepsilon$.

Application of LEMMA 3.9 yields the statement of this lemma. □

By virtue of this lemma we arrive at the main result of this paper:

THEOREM 3.11: *The imaginary stability boundary of a linear k-step method of order at least two, is at most* $(\frac{1}{2}\tilde{C}_3 + 1/3)^{-\frac{1}{2}}$, *where* $\tilde{C}_3$ *is the modified error constant of the third order term, defined by (2.4).*

PROOF: Let f(z,q) be the polynomial in z of degree k, associated with the k-step method, according to (3.1). Assume that f(z,q) is stable, for $q \in I$, $|q| < \beta$. Then, according to LEMMA 3.6, all roots of g(y,w), defined by (3.5) and (3.6), are real if $|w| < \beta$.
LEMMA 3.4 states that all roots of h_1 are non-negative, and according to REMARK 3.2, the positive roots are double. Denote these roots by $c_2,\ldots,c_m$, and consider the polynomial R(x,w) as defined by (3.11). We distinguish two cases:

(i) k is even. Then, we choose $h(x) = \sqrt{h_0(x)}$; the coefficient of the second term of h(x) is $-b_{k-2}$. Because $\sum_{j=2}^{m} c_j = -\frac{1}{2}a_{k-3}$, we have to take $c_1 = b_{k-2} - \frac{1}{2}a_{k-3} = -(\frac{1}{2}\tilde{C}_3 + 1/3)$ in order to satisfy (3.9).

(ii) k is odd. Then, we choose $h(x) = \sqrt{x\, h_0(x)}$, and again $c_1 = -(\frac{1}{2}\tilde{C}_3+1/3)^{-\frac{1}{2}}$.

In both cases, the conditions of LEMMA 3.10 are satisfied, if $c_1 < 0$; thus, $R(x,w_1)$ has at most 2m-2 real zeros, for a $w_1 \leq (-c_1)^{-\frac{1}{2}}+\varepsilon$.
However, $R(y^2,w)$ is equal to g(y,w) if k is even, and equal to $y^2 g(y,w)$ if k is odd. Thus, if k is even, $g(y,w_1)$ has at most 2(2m-2)=2k-4 real zeros; likewise, if k is odd, $g(y,w_1)$ has at most 2(2m-2)-2=2k-4 real zeros. Hence, $g(y,w_1)$ has complex zeros; however, all roots of g(y,w) are real if $|w| < \beta$. Thus, for all ε, $\beta < (-c_1)^{-\frac{1}{2}}+\varepsilon$, and we conclude that $\leq (-c_1)^{-\frac{1}{2}}$. □

COROLLARY 3.3: *The imaginary stability boundary of a linear k-step method of order higher than two, is at most* $\sqrt{3}$.

REFERENCES

[1] BARNETT, S. & D.D. SILYAK, *Routh's algorithm: a centennial survey*, SIAM Review 19, 472-489, 1977.

[2] CRYER, C.W., *A new class of highly stable methods: A_0-stable methods*, BIT 13, 153-159, 1973.

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

[4] DEKKER, K., *Formula manipulation in ALGOL 68 and application to Routh's algorithm*, Report 80-01, University of Amsterdam, 1980 (to appear in Computing).

[5] JELTSCH, R., *Stability on the imaginary axis and A-stability of linear multistep methods*, BIT 18, 170-174, 1978.

[6] JELTSCH, R. & O. Nevanlinna, *Stability of explicit time discretizations for solving initial value problems*, Report No 30, University of Oulu, 1979 (prepublication).

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

[8] MARDEN, M., *Geometry of polynomials*, Amer. Math. Soc., Providence, 1966.

[9] OBRESCHKOFF, N., *Verteilung und Berechnung der Nullstellen reeler Polynome*, VEB Deutscher Verlag der Wissenschaften, Berlin, 1963.

[10] VARAH, J.M., *Stiffly stable linear multistep methods of extended order*, Techn. Report 75-01, University of British Columbia, Vancouver, 1975.

SAMENVATTING

Vele in de natuur voorkomende verschijnselen kunnen beschreven worden door partiële differentiaalvergelijkingen, zoals bijvoorbeeld de één-dimensionale golfvergelijking

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \quad , \quad c > 0, \; 0 \leq x \leq 1, \; t \geq 0.$$

Indien de differentiaalvergelijking van een bepaald type is (bijv. lineair) en indien het gebied waarop de vergelijking gedefiniêerd is, een eenvoudige vorm heeft (bijv. een rechte lijn of een cirkel), dan kan men soms analytische methoden toepassen. Vaak zal men echter zijn toevlucht moeten nemen tot numerieke integratiemethoden. De basisideeën voor deze methoden zijn al aan het begin van deze eeuw ontwikkeld, maar de komst van snelle computers heeft pas tot een algemene toepassing van deze technieken geleid. Dit proefschrift bestaat uit vier wetenschappelijke publicaties,die met dit onderwerp samenhangen, en die worden aangeduid als de artikelen [A], [B], [C] en [D].

Een bekende methode om een partiële differentiaalvergelijking te herleiden tot een eenvoudiger probleem is de zogenaamde methode der lijnen. Dit is een discretiseringsmethode waarbij alleen de variabelen in de ruimterichting worden gediscretiseerd. Bij twee-dimensionale problemen wordt vaak, afhankelijk van de vorm van het definitiegebied, een kromlijnig rooster gebruikt. De afgeleiden worden dan benaderd door lineaire combinaties van functiewaarden in de roosterpunten (eindige differenties). In artikel A wordt een methode gegeven om de gewichten bij deze functiewaarden te bepalen, zodanig dat de discretisatiefout in zekere zin minimaal is. Hierdoor wordt de partiële vergelijking herleid tot een stelsel gewone differentiaalvergelijkingen, en wel één differentiaalvergelijking voor ieder roosterpunt. Als voorbeeld noemen we de ondiepwatervergelijkingen; deze beschrijven de beweging van lange golven in ondiepe zeeën. Het stelsel gewone differentiaalvergelijkingen dat hieruit ontstaat, heeft een hyperbolisch karakter, d.w.z. de eigenwaarden liggen in de buurt van de imaginaire as. Dit betekent dat de oplossing een golfbeweging vertoont; uiteraard willen we dat de numerieke methode die deze oplossing benadert, dit karakter handhaaft.

Bekende en veelvuldig toegepaste technieken voor het oplossen van stelsels gewone differentiaalvergelijkingen zijn de klassieke Runge-Kutta-formules en de klassieke meerstapsformules. In deze methoden wordt uitgaande van de oplossing op tijdstippen $t \leq t_n$ de oplossing op het tijdstip $t_{n+1}=t_n+h$ berekend. Om de integratie zo efficiënt mogelijk te laten verlopen, willen we de staplengte h graag zo groot mogelijk kiezen als op grond van de gewenste nauwkeurigheid toegelaten is. In de practijk blijkt echter vaak dat de numerieke oplossing onbegrensd toeneeemt, indien de staplengte groter gekozen wordt dan een bepaalde kritische waarde die stabiliteitsgrens genoemd wordt. Deze hangt niet slechts af van de eigenwaarden van het stelsel differentiaalvergelijkingen, maar ook van de gebruikte integratiemethode. In artikel [B] construeren we gegeneraliseerde Runge-Kuttamethoden. Hierbij zijn de parameters geen scalairen, maar diagonaalmatrices. In het bijzonder beschouwen we expliciete formules die geschikt zijn voor de integratie van semi-gediscretiseerde stelsels golfvergelijkingen. De toegelaten effectieve staplengte (dit is de staplengte gedeeld door het aantal benodigde functie-evaluaties) blijkt tweemaal zo groot als bij de klassieke Runge-Kuttamethoden mogelijk is.

In het geval van lineaire meerstapsmethoden wordt de stabiliteitsgrens bepaald door de voorwaarde dat alle nulpunten van het karakteristieke polynoom op of binnen de eenheidscirkel liggen. In artikel [C] geven we een implementatie van de algoritme van Routh, waarmee gecontroleerd kan worden of de nulpunten van een polynoom binnen de eenheidscirkel liggen. Met behulp van deze algoritme bewijzen we in artikel [D] dat de toegelaten staplengte voor methoden met een orde van nauwkeurigheid groter dan twee, begrensd wordt door $\sqrt{3}/\sigma$ waarbij σ de modulus van de grootste imaginaire eigenwaarde van het stelsel differentiaalvergelijkingen is.

STELLINGEN

1

Het polynoom $P(x) = 1 + x + x^2/2! + x^3/3! + x^4/4! + x^5/5! + .000725590420168\ x^6$ heeft de eigenschap $|P(x)| \leq 1$ als $-6.26 \leq x \leq 0$. BEENTJES en DEKKER geven een hierop gebaseerde 5e orde 6-punts Runge-Kuttaformule met de (op afronding na) optimale stabiliteitsgrens 6.26 .

BEENTJES, P.A. & K. DEKKER, *Een 5e orde, 6-punts Runge-Kuttaformule met optimale stabiliteitsgrens*, Rapport NR 27/72, Mathematisch Centrum, Amsterdam, 1972.

2

Alle nulpunten van het polynoom $g(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n$ liggen in het linker halfvlak, als er geen tekenwisselingen optreden in de rij $(s_{0,1}, s_{1,1}, \ldots, s_{n,1})$ die gedefiniëerd wordt door:

$$(s_0) = (a_0, a_2, a_4, \ldots), \qquad (s_1) = (a_1, a_3, a_5, \ldots),$$

$$s_{i+1,j} = -\begin{vmatrix} s_{i-1,1} & s_{i-1,j+1} \\ s_{i,1} & s_{i,j+1} \end{vmatrix}, \quad i=1,\ldots,n-1, \quad j=1,\ldots,(n+1-i)\div 2.$$

Als de coëfficiënten van g polynomen in z van de graad k ($k \geq 1$) zijn, dan neemt de bewerkelijkheid van deze delingsvrije variant van de algoritme van Routh sterk toe met de graad van g. De elementen $s_{i,j}$ zijn dan namelijk polynomen in z van de graad $k \times f_i$, waarbij f_i het i-de Fibonacci-getal is.

BARNETT, S. & D.D. SILYAK, *Routh's algorithm:a centennial survey*, SIAM Rev. 19, 472-489, 1977.

Dit proefschrift, pag. 52.

3

Volgens DAHLQUIST kan de orde van nauwkeurigheid van een lineaire meerstapsmethode voor differentiaalvergelijkingen van het type y" = f(t,y), die stabiel is langs de gehele negatieve reële as, niet groter zijn dan twee. Met behulp van de methode die in het vierde artikel van dit proefschrift beschreven wordt, kan worden aangetoond dat de maximale stabiliteitsgrens voor methoden van orde drie en hoger niet groter kan zijn dan 6.

DAHLQUIST, G., *On accuracy and unconditional stability of linear multistep methods for second order differential equations*, BIT 18, 133-136, 1978.

4

Het "odd-even hopscotch" schema, dat ontwikkeld is als directe methode voor partiële vergelijkingen, kan opgevat worden als een combinatie van een semi-discretiseringsmethode en een gegeneraliseerde (impliciete) Runge-Kutta-methode. De door VERWER gegeven vorm van het "odd-even hopscotch" schema vertoont grote overeenkomst met een Runge-Kuttaformule uit dit proefschrift. In de helft van de punten zijn beide schema's identiek.

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.

Dit proefschrift, pag. 37, formule (4.10).

5

Een automatisch stapkeuze strategie is een belangrijk onderdeel van procedures voor het oplossen van gewone differentiaalvergelijkingen. De extrapolatiemethode van BULIRSCH en STOER is volgens FOX zeer kostbaar indien de beginstap ongunstig gekozen wordt. Dit euvel wordt verholpen indien in de

procedure de regel "if j > 2 then konv:=true else konv:=false" vervangen wordt door "if j > 0 then konv:=true else konv:=false".

FOX, P., *A comparative study of computer programs for integrating differential equations*, Comm. ACM 15, 941-948, 1972.

BULIRSCH, R. & J. STOER, *Numerical treatment of ordinary differential equations by extrapolation methods*, Numer. Math. 8, 1-13, 1966.

6

Het is mogelijk met behulp van de procedure RK1N (ZONNEVELD) ook met weinig functie-evaluaties een redelijke benadering van de oplossing van het drie-lichamenprobleem te berekenen, en wel door de tolerantie per integratiestap constant te kiezen, en niet evenredig met de staplengte.

ZONNEVELD, J.A., *Automatic numerical integration*, MC Tracts 8, Mathematisch Centrum, Amsterdam, 1970.

7

Een oplegprobleem wordt beschreven door de Fredholm-vergelijking van de tweede soort (HERREBRUGH)

$$f(x) = f(0) + \lambda \int_0^b K(x,y)\ f(y)\ dy, \qquad 0 \le x \le b,$$

waarin λ een vrije parameter is, en K gedefiniëerd wordt door

$$K(x,y) = \begin{cases} y^2\ (y - 3x), & 0 \le y \le x, \\ x^2\ (x - 3y), & x \le y \le b, \end{cases}$$

onder de nevenvoorwaarde

$$\int_0^b f(y)\ dy = 1.$$

Deze integraalvergelijking is equivalent met het randwaardeprobleem

$$f''''(x) = -6\lambda f(x), \quad 0 \le x \le b, \quad f(b)=f''(b)=f'''(b)=0,$$
$$f'(0)=0, \quad f'''(0)=6\lambda.$$

De analytische oplossing van deze vergelijking luidt

$$f(x) = -\frac{\alpha}{\sinh \frac{1}{2}\pi}\{\cosh\alpha(x-b)\sin\alpha(x-b) + \sinh\alpha(x-b)\cos\alpha(x-b)\},$$

waarin

$$\alpha = (3\lambda/2)^{\frac{1}{4}}.$$

HERREBRUGH, K., *Een oplegprobleem*, Werkgroep differentiaal- en integraalvergelijkingen, Mathematisch Centrum, Amsterdam, 1978.

8

In een veem wordt een bepaald artikel opgeslagen. De klantenorders voor dit produkt komen binnen volgens een Poisson proces met parameter λ; de grootte van de orders is exponentiëel verdeeld met parameter ν. Indien aan een order niet voldaan kan worden, dan wordt deze nageleverd. Bij een ander veem kan het artikel besteld worden; er mag slechts één bestelling uitstaan, en de levertijd is exponentiëel verdeeld met parameter μ. De gemiddelde kosten per tijdseenheid bedragen voor een (S,s)-strategie:

$$Y = d\frac{\lambda}{\nu} + c\mu\xi + a\{S - \frac{\lambda}{\mu\nu} - \xi\left(\frac{\mu\nu\Delta}{2\lambda} + \left(\frac{\lambda\Delta}{\lambda+\mu} + \frac{\lambda}{\mu\nu}\right)e^{-\frac{\mu\nu\Delta}{\lambda+\mu}}\right)\} +$$
$$+ (a+b)\,\xi\, e^{-\frac{\mu\nu S}{\lambda+\mu}}\left\{(S-\Delta)\frac{\lambda^2}{(\lambda+\mu)^2} + \frac{\lambda^2}{\mu\nu(\lambda+\mu)} + \frac{\lambda+\mu}{\mu\nu}e^{\frac{\mu\nu\Delta}{\lambda\mu}}\right\},$$

waarin a de voorraadkosten per tijdseenheid zijn, b de naleveringskosten per tijdseenheid, c+dx de bestelkosten voor een bestelling x, en ξ en Δ gegeven worden door

$$\xi = \left(\frac{\mu}{\lambda} + \frac{\mu}{\lambda}\nu\Delta + \frac{\lambda}{\lambda+\mu}e^{-\frac{\mu\nu\Delta}{\lambda+\mu}}\right)^{-1}, \quad \Delta = S - s.$$

DEKKER, K., *Een continu voorraadprobleem met levertijden* Rapport BN 17/72, Mathematisch Centrum, Amsterdam, 1972.

9

Als $h(x)$ kwadratisch integreerbaar is over elk eindig interval, en als voor alle $p > 0$ geldt $\lim_{x \to -\infty} e^{px}\, h(x) = 0$, dan wordt de oplossing van de Volterra-vergelijking

$$f(x) = h(x) + \alpha \int_{-\infty}^{x} \nu\, e^{-\nu(x-y)}\, f(y)\, dy, \quad \alpha < 1,$$

die voldoet aan de voorwaarde $\lim_{x \to -\infty} e^{px}\, f(x) = 0$ voor alle $p > 0$, gegeven door

$$f(x) = h(x) + \alpha \int_{-\infty}^{x} \nu\, e^{-(1-\alpha)(x-y)}\, h(y)\, dy.$$

Met behulp van deze transformatie kan het stelsel integraalvergelijkingen waartoe het probleem uit de voorgaande stelling aanleiding geeft, opgelost worden.

10

Een strategie voor het bepalen van een pivot van een vierkante matrix A van orde n luidt: zoek een element $A_{k,m}$ zodanig dat geldt (DEKKER)

$$|A_{k,m}| = \max_i |A_{i,m}| = \max_j |A_{k,j}| .$$

Indien de elementen van A random gekozen worden uit een homogene verdeling, dan kan een element dat aan bovengenoemde voorwaarde voldoet, voor grote waarde van n met gemiddeld e maal n vergelijkingen gevonden worden.

DEKKER, T.J., *Pivotstrategieen*, Interne notitie, Universiteit van Amsterdam, 1977.

11

De procedure multiplypolbypol (TEER) geeft in sommige gevallen een onjuist resultaat. De pointer 13 blijft namelijk na het verwijderen van een term nog naar deze term verwijzen.

TEER, F., *Polynomial manipulation in ALGOL 68*, Rapport IR-28, Vrije Universiteit, Amsterdam, 1978.

12

In het ratingsysteem van ELO wordt aan iedere schaker een (geheel) getal toegekend dat zijn speelsterkte aangeeft. Aangenomen wordt dat de prestaties van een speler normaal verdeeld zijn met een spreiding $\sigma=200$. De winstverwachting in een partij luidt dan

$$E(s) = \frac{1}{2\sigma\sqrt{\pi}} \int_{-\infty}^{s} e^{-\frac{1}{4}(t/\sigma)^2}\, dt,$$

waarin s het krachtsverschil met de tegenstander aangeeft. In feite gebruikt Elo echter de functie

$$\tilde{E}(s) = \tfrac{1}{2}\, \mathrm{erf}\left(\frac{7\sqrt{2}}{1000}\, s\right) + \tfrac{1}{2}.$$

Op een zakrekenmachine kan deze functie met een nauwkeurigheid van vier cijfers benaderd worden met de algoritme

$$y := \frac{s}{358}\,, \quad E := \tfrac{1}{2}y\, \frac{1 + 0.01875\, y^2}{1 + 0.29136\, y^2} + \tfrac{1}{2}.$$

ELO, A.E., *The rating of chessplayers, Past & Present*, B.T.Batsford, London, 1978.

13

Laat A een symmetrische vierkante (ijle) matrix zijn waarvan de elementen de aantallen door twee spelers onderling gespeelde partijen aangeven, en $\vec{b}$ een vector die de scores van de spelers bevat. De oplossing $\vec{x}$ van het stelsel niet-lineaire vergelijkingen ($\tilde{E}$ wordt gegeven in stelling 12)

$$\sum_j A_{i,j}\, \tilde{E}(x_i - x_j) = b_i, \quad i=1,\ldots,n,$$

bevat de rating van de spelers, op een willekeurig te kiezen constante na. Het verdient aanbeveling de gevoeligheid van de oplossing voor verstoringen van de vector $\vec{b}$ te analyseren.

14

De eindspelbehandeling van veel schaakprogramma's staat op een bedroevend peil. Hierin kan verbetering gebracht worden door algemene principes te hanteren, in plaats van eindeloos varianten door te rekenen. Met behulp van de theorie der toegevoegde velden (CHERON) kan bijvoorbeeld het pionneneindspel: Wit: Ke5, c4, a3, Zwart: Ka5, a4, c5, c6 (Wit aan zet), gemakkelijk met een computer opgelost worden.

LEVY, D., *Chess and computers*, B.T. Batsfort, London, 1976.

CHERON, A., *Lehr- und Handbuch der Endspiele II*, Siegfried Engelhardt Verlag, Berlin, 1964.

15

De speelsterkte van een schaakprogramma wordt niet alleen bepaald door de vaardigheid van de ontwerpers als programmeur, maar ook door hun kennis van het schaakspel. Dit laatste wordt onvoldoende onderkend.

16

Het verdient aanbeveling om in grote wetenschappelijke centra een sportzaal op te nemen.