stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE NW 5/73 JULY

J.G. VERWER
A TWO-STEP RUNGE-KUTTA METHOD OF THIRD
ORDER WITH EXTENDED STABILITY REGION

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS (MOS) subject classification scheme (1970): 65L05

Abstract

In this report a two-step Runge-Kutta method of third order is proposed which is based on three function evaluations. This two-step formula is compared with the classic third order formula of Heun. An ALGOL 60 procedure is described which implements both formulae. Numerical experiences are reported.

Contents

1. Introduction

Both one- and two-step Runge-Kutta methods may be used to solve numerically initial value problems for systems of ordinary differential equations of the type

$$\frac{dU}{dt} = H(t,U) \ .$$

Up to now the one-step methods are more widely used. The two-step Runge-Kutta methods were first considered by Byrne and Lambert (see reference [6]). The two-step scheme discussed here was given by v.d. Houwen (see [1]). This scheme is closely related to the usual one-step scheme, since it does not use preceding evaluations of H(t,U).

Some theoretical aspects of the method developed are discussed in chapter 2. Special attention will be paid to stability.

In chapter 3 is given a third order exact formula which uses three evaluations of H(t,U). The real stability boundary of this formula varies between 4.3 and 5, whereas the real stability boundary of the corresponding one-step formula is 2.5.

In chapter 4 we present the procedure two step runge kutta. This procedure is an ALGOL 60 realization of the third order formula given in chapter 3.

The last chapter presents some results of a comparative analysis of the one- and two-step methods.

2 An explicit two-step Runge-Kutta method

In this chapter we present some theoretical aspects of the method developed. The stability of the method will have our special attention.

2.1 General structure of the integration scheme

Consider the initial value problem

$$(2.1)\quad \begin{cases} \dfrac{dU}{dt} = H(t,U), & t \geq t_0 \ , \\ U = \tilde{U}_0 \ , & t = t_0 \ , \end{cases}$$

where $\tilde{U}_0$ is a given initial vector and H is a given (vector) function of t and U. The analytical solution of (2.1) will be denoted by $\tilde{U}$. Suppose H has derivatives with respect to t and U of sufficiently high order.

Our two-step scheme has the following form (see [1])

$$(2.2)\quad \begin{cases} U_0 = \tilde{U}_0, \\ U_{k+1} = \gamma[U_k + \theta_0\, r_k^{(0)} + \ldots + \theta_{n-1}\, r_k^{(n-1)}] + (1-\gamma)U_{k-1}, \\ r_k^{(0)} = \tau_k H(t_k, U_k), \\ \quad \ldots \\ r_k^{(j)} = \tau_k H(t_k+\mu_j\tau_k, U_k+\lambda_{j0} r_k^{(0)} + \ldots + \lambda_{jj-1} r_k^{(j-1)}), \\ \quad \ldots \\ r_k^{(n-1)} = \tau_k H(t_k+\mu_{n-1}\tau_k, U_k+\lambda_{n-10} r_k^{(0)} + \ldots + \lambda_{n-1\,n-2}\, r_k^{(n-2)}), \\ k = 1,2,\ldots \ , \end{cases}$$

where we assume U_1 has been calculated with some one-step method. In this scheme we have

U_k: numerical approximation to the analytical solution $\tilde{U}$ at $t = t_k$,

τ_k: the steplength $t_{k+1} - t_k$,

γ, θ_j, μ_j, λ_{jl}: real parameters to be determined by consistency and stability conditions.

The parameter γ will be of importance for the stability of the method. It will be called the *stability parameter*. Note that for $\gamma = 1$ scheme (2.2) reduces to the usual one-step scheme.

Scheme (2.2) may be characterized by γ and the matrix

$$(2.3)\quad R = \left[\begin{array}{c|cccc} 0 & & & & \\ \mu_1 & \lambda_{10} & & & \\ \mu_2 & \lambda_{20} & \lambda_{21} & & \\ \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & & \cdot \\ \mu_{n-1} & \lambda_{n-10} & \lambda_{n-11} & \cdots & \lambda_{n-1n-2} \\ \hline & \theta_0 & \theta_1 & \cdots \theta_{n-2} & \theta_{n-1} \end{array}\right]$$

2.2 Consistency conditions

Here, we introduce a set of parameters β_j which are convenient in the formulation of consistency and stability (compare [2], formula (3.6)). They are defined by

$$(2.4)\quad \left\{\begin{array}{l} \beta_1 = \sum_{j=0}^{n-1} \theta_j \,, \\ \beta_2 = \sum_{j=1}^{n-1} \theta_j \mu_j \,, \\ \beta_3 = \sum_{j=2}^{n-1} \theta_j \sum_{l=1}^{j-1} \lambda_{jl}\mu_l \,, \quad \beta_{31} = \sum_{j=1}^{n-1} \theta_j \mu_j^2 \,, \\ \quad \cdot \\ \quad \cdot \\ \quad \cdot \end{array}\right.$$

Furthermore, we introduce the growthparameter

$$(2.5)\qquad c_k = \frac{\tau_{k-1}}{\tau_k} \,, \quad k = 1,2, \ldots \,.$$

It will be assumed that both c_k and $1/c_k$ are uniformly bounded with respect to k. Henceforth we omit the index k. Moreover, we assume throughout this paper that the following relations are satisfied:

$$(2.6) \qquad \sum_{l=0}^{j-1} \lambda_{jl} = \mu_j \ , \quad j = 1,2,\ldots,n-1 \ .$$

These relations simplify the calculations (compare [4]).

Let the integration method (2.2) be written as

$$U_{k+1} = L_k(U_{k-1},U_k) \ .$$

Then *the order of consistency* is defined in the following way:

Definition 2.1

The method is said to be consistent of order p at the point $t = t_k$ if

$$U(t_{k+1}) - L_k(U(t_{k-1}), U(t_k)) = O(\tau_k^{p+1}) \text{ as } \tau_k \to 0 \ ,$$

where U(t) is the solution of the differential equation which satisfies the condition $U(t_k) = U_k$.

Theorem 2.1

The integration method (2.2) is of order p = 1 if

$$(2.7) \qquad \beta_1 = (1+(1-\gamma)c)/\gamma \ ,$$

of order p = 2 if, in addition,

$$(2.8) \qquad \beta_2 = (1-(1-\gamma)c^2)/2\gamma \ ,$$

of order p = 3 if, in addition,

$$(2.9) \qquad \beta_3 = (1+(1-\gamma)c^3)/6\gamma \ ,$$

$$(2.10) \qquad \beta_{31} = (1+(1-\gamma)c^3)/3\gamma \ .$$

Proof

The proof can be given by a Taylor expansion of $L_k(U(t_{k-1}),U(t_k))$ with respect to τ_k. By identifying the first p+1 terms of this expansion with the Taylor series of U(t), we obtain the conditions for p-th order consistency (compare [2], section 3.2, and [4]). In order to guarantee convergence, we assume that $\beta_1 \neq 0$ (see [1], p.4).

2.3 Stability

We shall investigate the stability of the integration method as applied to the linear differential equation

$$\frac{dU}{dt} = DU + F \quad , \tag{2.11}$$

where D is a matrix with constant entries and F is a (vector) function of the variable t.

This approach is suggested by the fact that locally each non-linear differential equation

$$\frac{dU}{dt} = H(t,U) \quad , \tag{2.12}$$

has the form (2.11), where D is the Jacobian of H(t,U) (compare [3], p.2). Stability of this kind is called *linear- or local stability*.

2.3.1 The error of the difference scheme

In each integration step there are two types of errors:

1. *The local discretization error* - the error introduced by approximating the differential equation by a difference equation
2. *The local numerical error* - the error introduced by round-off errors which give rise to a numerical solution U^* instead of the difference solution U.

During the integration process, both types of errors accumulate in a so-called *global error*. A difference scheme for the global error will be

presented below. It is easily verified that in the case of linear differential equations (2.11), scheme (2.2) reduces to the scheme

$$U_{k+1} = \gamma P_n(\tau_k D) + (1-\gamma)U_{k-1} + \tau_k\, g_k^{(n)} \quad , \quad k = 1,2\ldots, \tag{2.13}$$

where

$$P_n(z) = 1 + \beta_1\, z + \beta_2\, z^2 + \ldots + \beta_n\, z^n \quad , \tag{2.14}$$

and where the vector $g_k^{(n)}$ is determined by the vectors $F(t_k+\mu_j t)$. For $\gamma = 1$, it has been proved, in reference [3], section 2.1-2.3, that the global error

$$e_k = \tilde{U}(t_k) - U_k^* \tag{2.15}$$

satisfies the difference scheme

$$e_{k+1} = P_n(\tau_k D)e_k + s_k \quad , \quad k = 0,1,2,\ldots, \tag{2.16}$$

where s_k is the sum of the local discretization error and the local numerical error.

For all γ, it can be proved, in a completely analogous way, that the global error (2.15) satisfies the difference scheme

$$e_{k+1} = \gamma P_n(\tau_k D)e_k + (1-\gamma)e_{k-1} + s_k \quad , \quad k = 1,2,\ldots \ . \tag{2.17}$$

2.3.2 Stability and eigenvalues

With scheme (2.17), we shall demonstrate that the stability of the integration method depends on the eigenvalues of the matrix D.

Suppose D has an orthogonal system of eigenvectors $\{E_j\}$ with eigenvalues δ_j. Then the matrix $P_n(\tau_k D)$ has the same orthogonal system of eigenvectors $\{E_j\}$ with eigenvalues $P_n(\tau_k \delta_j)$. Consequently, we may write

$$(2.18) \qquad e_k = \sum_j e_k^{(j)} E_j \ , \ s_k = \sum_j s_k^{(j)} E_j \ .$$

After substitution of (2.18) into (2.17) we obtain

$$(2.19) \qquad \sum_j [e_{k+1}^{(j)} - \gamma P_n(\tau_k \delta_j) e_k^{(j)} - (1-\gamma) e_{k-1}^{(j)} - s_k^{(j)}] . E_j = 0, \quad k = 1,2,\ldots \ .$$

The system $\{E_j\}$ is independent, so for each j we have

$$(2.20) \qquad e_{k+1}^{(j)} = \gamma P_n(\tau_k \delta_j) e_k^{(j)} + (1-\gamma) e_{k-1}^{(j)} + s_k^{(j)} \ , \quad k = 1,2,\ldots \ .$$

The error vector e_k will not grow in any norm, if for each j the absolute value of $e_k^{(j)}$ does not increase. Therefore we continue with the single difference equation (2.20).

In general the homogeneous solution of (2.20) determines the accumulated error at each step (compare [5], section 3.2). It thus seems reasonable to approximate the inhomogeneous equation (2.20) with *the homogeneous error equation.*

$$(2.21) \qquad \varepsilon_{k+1} = \gamma P_n(z) \ \varepsilon_k + (1-\gamma) \ \varepsilon_{k-1} \ , \quad k = 1,2,\ldots \ ,$$

where $z = \tau_k \delta$ and δ is some eigenvalue of D.

Thus we have reduced scheme (2.17) to the single error equation (2.21). From this we may conclude that in case of a normal matrix D the local stability of the integration method (2.2) depends on the eigenvalues of D.

When the integration method (2.2) is applied to the single linear differential equation

$$(2.22) \qquad \frac{dU}{dt} = \delta U \ ,$$

scheme (2.13) is reduced to

$$(2.23) \qquad U_{k+1} = \gamma P_n(z) \ U_k + (1-\gamma) \ U_{k-1} \ , \quad k = 1,2,\ldots \ ,$$

and scheme (2.17) is reduced to (2.21). Therefore it is sufficient to consider the single linear equation (2.22). In case of a non-normal matrix D this approach is applied too.

2.3.3 Absolute and relative stability

Suppose we have integrated (2.22) up to the point $t = t_{k_0}$ and $\tau_k = \tau$, $k = k_0, k_0 + 1, \ldots$. The difference equations (2.21) and (2.23) are thus reduced to

$$\varepsilon_{k+1} = \gamma P_n(z)\, \varepsilon_k + (1-\gamma)\, \varepsilon_{k-1}\,, \qquad k = k_0\,, k_0+1, \ldots\,, \tag{2.24}$$

$$U_{k+1} = \gamma P_n(z)\, U_k + (1-\gamma)\, U_{k-1}\,, \qquad k = k_0\,, k_0+1, \ldots\,, \tag{2.25}$$

where $z = \tau\delta$. The *characteristic roots* of the linear difference equation with constant coefficients (2.25) are the roots of the characteristic equation

$$\lambda^2 - \gamma P_n(z)\lambda - (1-\gamma) = 0. \tag{2.26}$$

If the roots are distinct, the solution of (2.25) takes the form

$$U_k = \alpha_1 \lambda_1^k + \alpha_2 \lambda_2^k\,, \qquad k = k_0\,, k_0+1, \ldots\,, \tag{2.27}$$

where α_1 and α_2 are constants determined by the initial conditions. If the roots are equal the solution of (2.25) takes the form

$$U_k = \alpha_1 \lambda_1^k + \alpha_2 k \lambda_1^k\,, \qquad k = k_0\,, k_0+1, \ldots\,. \tag{2.28}$$

One of the characteristic roots approximates the analytical solution $C_k e^{\delta\tau}$ of (2.22). This root is called *the principal root* and is denoted by λ_1. The remaining root λ_2 is called *the parasitic root*, and arises because a second-order difference equation has been used to approximate a first-order differential equation. The parasitic root has no relation to the exact

solution of the differential equation.

The characteristic roots of the difference equation for the error ε_k are the same as those of (2.25). Therefore the solution of (2.24) also takes the form (2.27) or (2.28). Thus, at each integration step k_0 the stability is determined by the roots of (2.26) and is defined in the following way (compare [5], section 3.2)

Definition 2.2

The integration method (2.2) is called

absolute stable if $|\lambda_i| \leq 1$, $i = 1,2$,

relatively stable if $|\lambda_2| \leq |\lambda_1|$.

Let us be more specific with regard to the absolute stability. Suppose $0 \leq \gamma \leq 2$. Substitute $\lambda = e^{i\phi}$ in (2.26) and solve $P_n(z)$. This results in

$$(2.29) \qquad P_n(z) = e^{-i\phi} + \frac{2i}{\gamma} \sin\phi \ , \ 0 \leq \phi \leq 2\pi \ , \ 0 \leq \gamma \leq 2 \ .$$

The curve δS in the complex Z-plane defined by (2.29) is called *the general boundary of absolute stability*. The domain S bounded by δS is called *the region of absolute stability*. At each integration step k_0 we have absolute stability if the point $z = \tau_{k_0} \delta$ belongs to S. The region S is symmetric with respect to the real z-axis.

2.3.4 Negative eigenvalues

In the case of negative eigenvalues, the characteristic equation (2.26) has real coefficients.

First we give the condition which determines *the real boundary of absolute stability*, β_{real}; and secondly, the condition which determines *the real boundary of relative stability*, α_{real}.

1. The characteristic roots are within or on the unit circle if

$$(2.30) \qquad |P_n(z)| \leq 1 \ , \quad 0 \leq \gamma \leq 2 \ .$$

In the next chapter γ will be used to maximize β_{real}.

2. The principal root $\lambda_1 \to 1$ as $\tau_k \to 0$. Therefore, we have

$$(2.31) \quad \begin{cases} \lambda_1 = \frac{1}{2}\gamma P_n(z) + \frac{1}{2}\sqrt{D(z)} \ , \\ \lambda_2 = \frac{1}{2}\gamma P_n(z) - \frac{1}{2}\sqrt{D(z)} \ , \end{cases}$$

where $D(z) = \gamma^2 P_n^{\,2}(z) + 4 - 4\gamma$. Let us distinguish two possibilities:

a. $D(z) \leq 0$:

Here we have $|\lambda_1| = |\lambda_2|$. This implies relative stability.

b. $D(z) > 0$:

In this case it is easily verified that

$$(2.32) \qquad |\lambda_2| \leq |\lambda_1| \iff P_n(z) \geq 0 \ .$$

Consequently, α_{real} is the first zero of $P_n(z)$ on the negative z-axis.

3. A third order exact scheme

In this chapter we present the scheme of which an Algol 60 version can be found in chapter 4.

3.1. The generating matrix

In scheme (2.2) we choose n=3. The scheme then uses three evaluations of H(t,U). With the relations (2.4) and (2.6) we find expressions for the parameters θ_j, μ_j and λ_{j1}. In order to simplify the difference scheme we substitute $\theta_1 = \lambda_{20} = 0$. Calculations then yield

$$(3.1)\quad R = \left[\begin{array}{c|ccc} 0 & & & \\ \frac{\beta_3}{\beta_2} & \frac{\beta_3}{\beta_2} & & \\ \frac{2\beta_3}{\beta_2} & 0 & \frac{2\beta_3}{\beta_2} & \\ \hline & \beta_1 - \frac{\beta_2^2}{2\beta_3} & 0 & \frac{\beta^2}{2\beta_3} \end{array}\right]$$

If the parameters β_j are defined by the consistency conditions (2.7)-(2.9), R generates a third order exact scheme. The elements of R satisfy condition (2.10).

The generating matrix of the corresponding one-step scheme is (compare [5], section 2.3.3)

$$(3.2)\quad R = \left[\begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}\right]$$

3.2 Regions of absolute stability

The general boundary of absolute stability δS is defined by

$$(3.3) \qquad P_3(z) = e^{-i\phi} + \frac{2i}{\gamma}\sin\phi \ , \ 0 \leq \phi \leq 2\pi \ , \ 0 \leq \gamma \leq 2 \ ,$$

where

$$(3.4) \qquad P_3(z) = 1 + \frac{1 + (1-\gamma)c}{\gamma} z + \frac{1 - (1-\gamma)c^2}{\gamma}\frac{z^2}{2} + \frac{1 + (1-\gamma)c^3}{\gamma}\frac{z^3}{6} \ .$$

Obviously, each value of the growthparameter c determines a δS. To show this dependence of δS on c, we have illustrated some stability regions in fig. 3.1-3.5. The choice of the parameters c and γ will be clear after the next section.

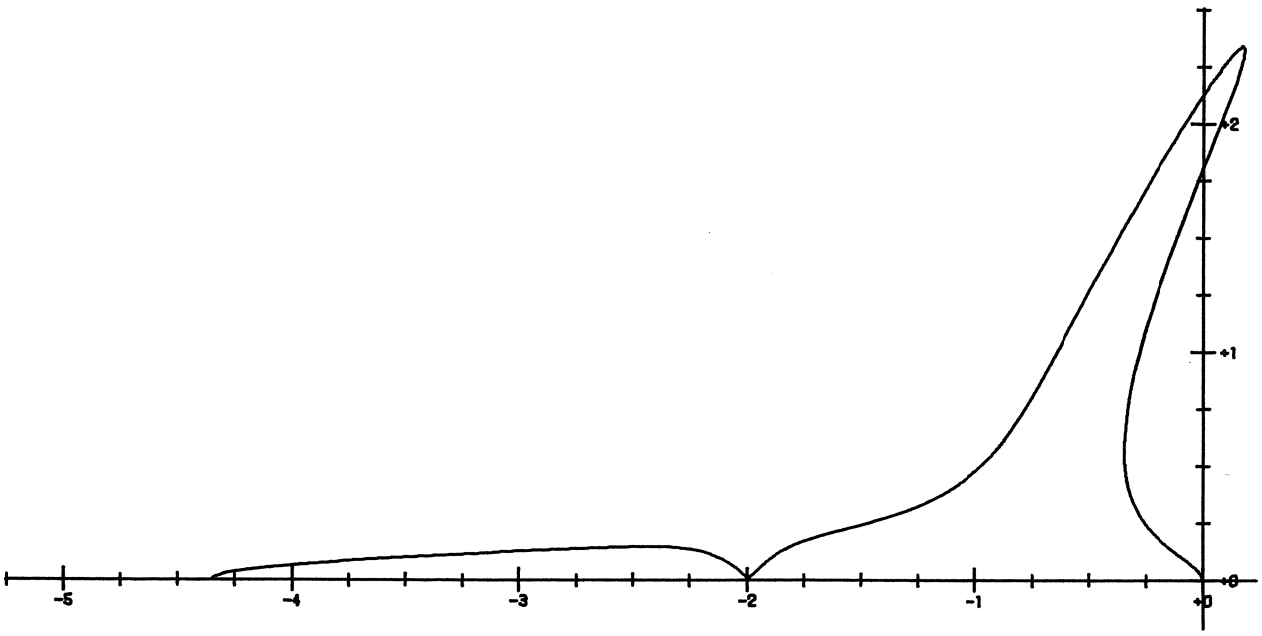

Fig. 3.1 δS for c = .5, $\gamma = \gamma_R(c)$.

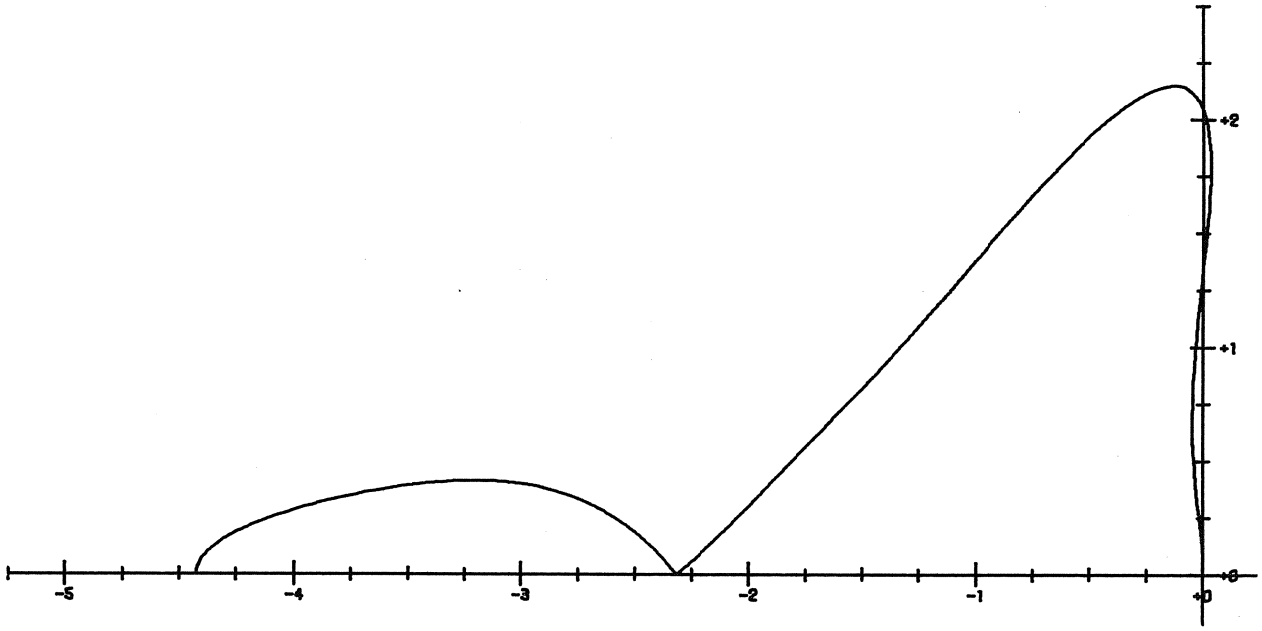

Fig. 3.2 δS for $c = .8$, $\gamma = \gamma_R(c)$.

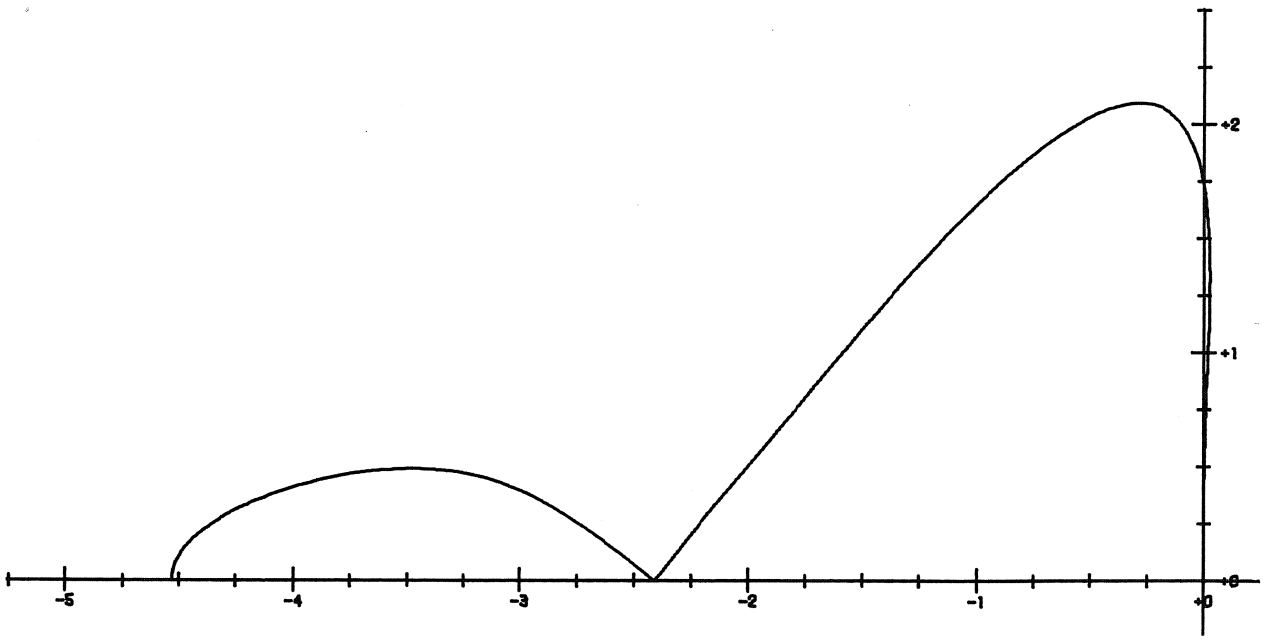

Fig. 3.3 δS for $c = 1$, $\gamma = \gamma_R(c)$.

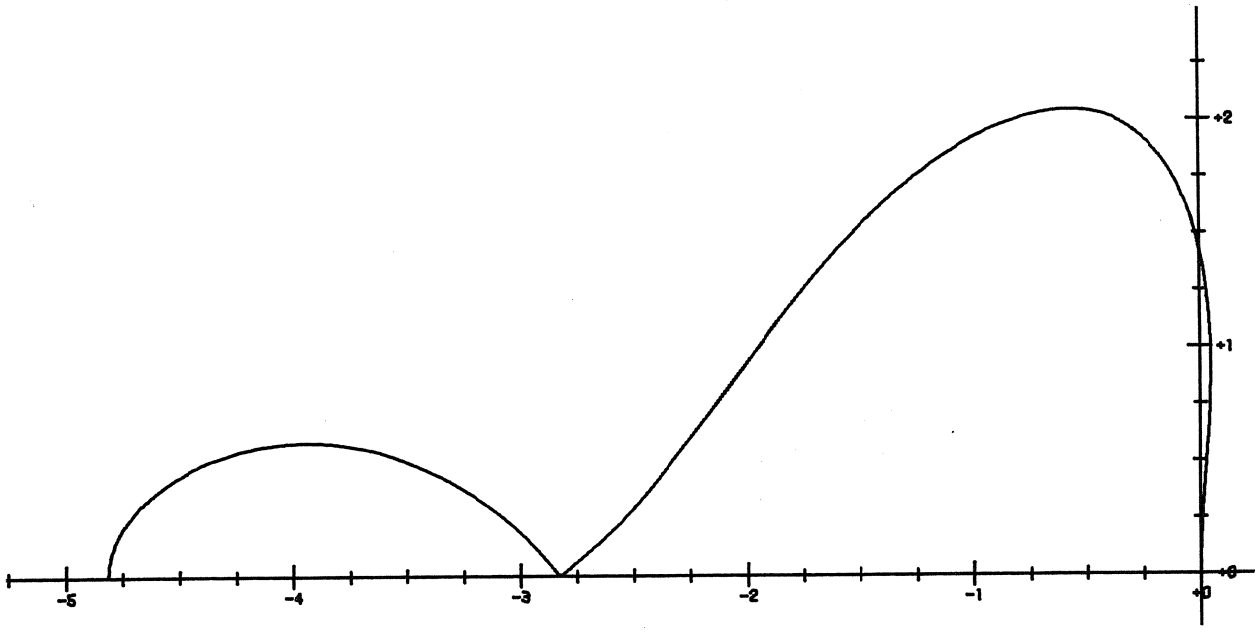

Fig. 3.4 δS for $c = 1.5$, $\gamma = \gamma_R(c)$.

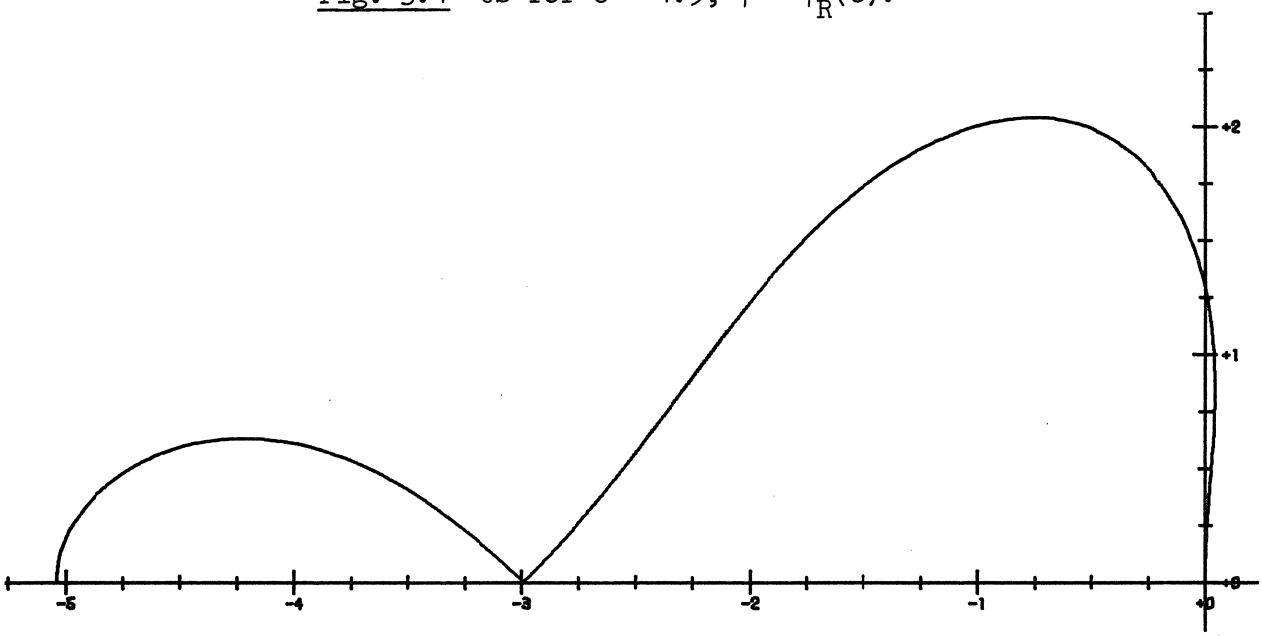

Fig. 3.5 δS for $c = 2$, $\gamma = \gamma_R(c)$.

3.3 Stability in case of negative eigenvalues

In this section δ is assumed to be negative. We shall concentrate on absolute stability. Let us define the polynomials

$$(3.5)\qquad \begin{cases} T(z) = c - \dfrac{c^2}{2} z + \dfrac{c^3}{6} z^2 , \\ N(z) = (c+1) - \dfrac{c^2-1}{2} z + \dfrac{c^3+1}{6} z^2 . \end{cases}$$

Now condition (2.30) with n = 3 can be reduced to

$$(3.6)\qquad \frac{T(z)}{N(z)} \le \frac{1}{\gamma} \le \frac{zT(z) - 2}{zN(z)} , \qquad 0 \le \gamma \le 2 .$$

At this point we use the fact that the parameter γ is still a free parameter. The problem is to determine γ in such a way that the interval $-\beta_{real} \le z \le 0$ is as large as possible, while satisfying condition (3.6). For c = 1, or constant stepsize, (3.6) is reduced to

$$(3.7)\qquad Y_1(z) \le \frac{2-\gamma}{\gamma} \le Y_2(z) , \quad 0 \le \gamma \le 2 ,$$

where

$$(3.8)\qquad Y_1(z) = \frac{-z^2}{2z(1 + \frac{z^2}{6})} , \; Y_2(z) = -\frac{4 + z^2}{2z(1 + \frac{z^2}{6})} .$$

In fig. 3.6, $Y_1(z)$ and $Y_2(z)$ are illustrated. The optimal choice of γ is defined by

$$(3.9)\qquad \frac{2 - \gamma}{\gamma} = Y_1(z_0) ,$$

where z_0 satisfies the equation

$$(3.10)\qquad \frac{d}{dz} Y_1(z) = 0 .$$

The optimal choice of β_{real} is defined by

$$(3.11)\qquad Y_2(-\beta_{real}) = Y_1(z_0) .$$

A simple calculation yields

(3.12) $\quad z_0 = -\sqrt{6}\ , \quad \gamma = \dfrac{8}{4+\sqrt{6}}\ , \quad \beta_{real} \simeq 4.5\ .$

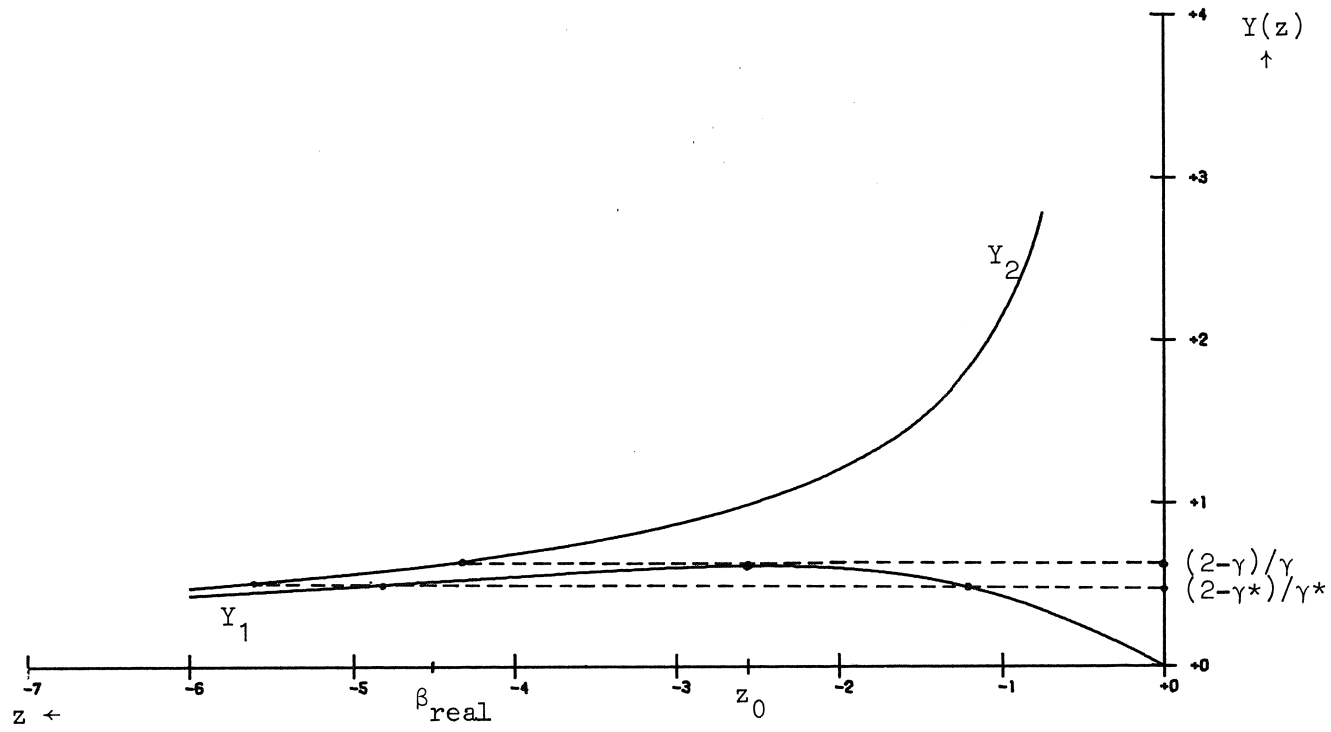

Fig. 3.6.

With the help of fig. 3.6, we shall derive in an heuristic way an expression for *the optimal value of* γ for all values of c. If in fig. 3.6 $\gamma = 8/(4+\sqrt{6})$ or $\gamma = \gamma^*$, δS has respectively three or four points on the real z-axis. These points represent real roots of equation (3.3). Evidently in the optimal case two real roots coincide. Only if $\phi = 0$ or $\phi = \pi$ equation (3.3) does have one or three real roots. Let us try $\phi = 0$. In this case (3.3) reduces to

(3.13) $\quad \beta_1 z + \beta_2 z^2 + \beta_3 z^3 = 0\ .$

This equation has two equal roots if

$$(3.14) \qquad \beta_2^2 - 4\beta_1\beta_3 = 0 \ .$$

Solving (3.14) for γ, we have

$$(3.15) \qquad \gamma_R(c) = \frac{M(c) + 2c^4 - \sqrt{M^2(c) - 4c^4}}{2c^4} \ ,$$

where

$$(3.16) \qquad M(c) = \frac{8}{5} c^3 + \frac{6}{5} c^2 + \frac{8}{5} c \ .$$

From numerical experiments with condition (3.6), it appears that $\gamma = \gamma_R(c)$ does represent the optimal value of γ in relation to the optimal choice of β_{real}. The function $\gamma_R(c)$ has been illustrated in fig. 3.7. To satisfy the condition $0 \leq \gamma \leq 2$, the growthparameter c must be limited to the interval $c \gtrsim .4$.

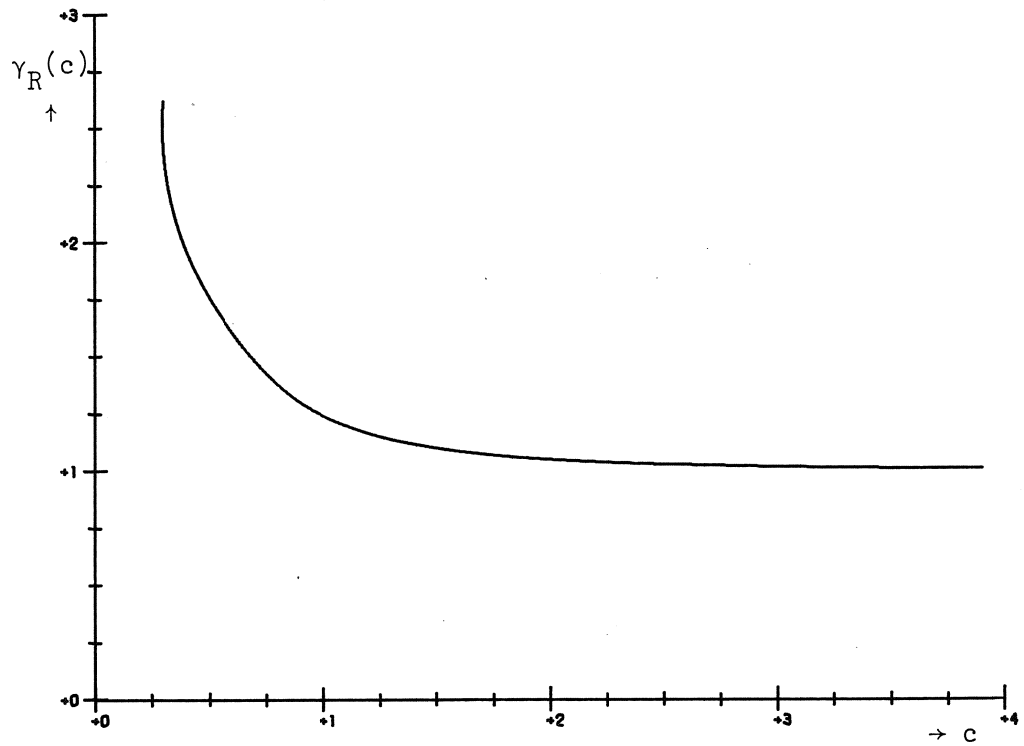

Fig. 3.7 The stabilityparameter $\gamma_R(c)$.

The numerical experiments with (3.6), yielded the β_{real}'s as given in table 3.1. The corresponding α_{real}'s are numerically determined as the real root of

$$(3.17) \qquad P_3(z) = 0 \ , \quad \gamma = \gamma_R(c) \ ,$$

and are also given in table 3.1.

Table 3.1. Real stability boundaries

c	β_{real}	α_{real}
.4	4.3	4.3
.5	4.3	4.3
.7	4.3	4.3
.9	4.4	4.4
1.0	4.5	4.4
1.2	4.6	4.5
1.4	4.7	4.5
1.6	4.8	4.6
1.8	4.9	4.7
2.0	5.0	4.7

As long as $.4 \leq c \leq 2$, the minimum value of β_{real} and α_{real} appears to be 4.3. The real stability boundary of the corresponding one-step method is 2.5 (compare [3], section 3.1). Consequently we have an increase of the stability interval of about 42%.

Finally, we have illustrated the absolute value of the characteristic roots in fig. 3.8 for c = 1. The characteristic roots are given by (2.31), with n = 3 and $\gamma = 8/(4+\sqrt{6})$.

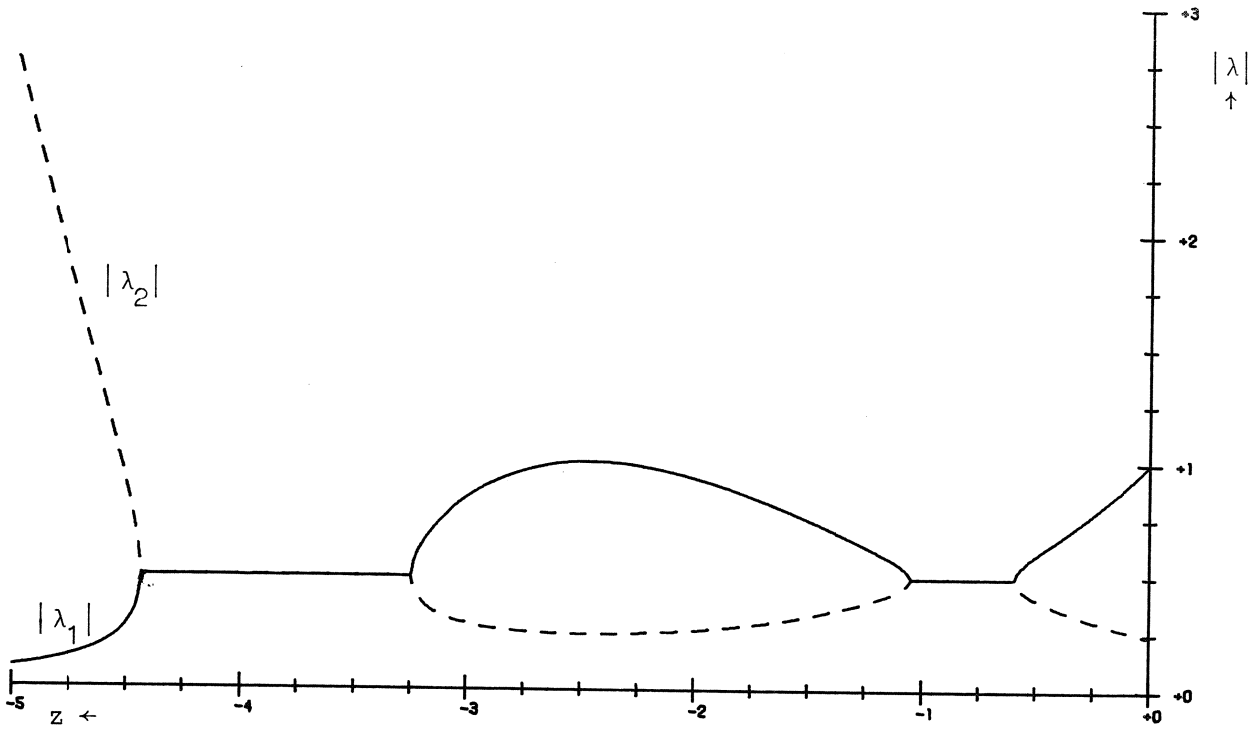

Fig. 3.8 The characteristic roots.

3.4 Stability in case of non-real eigenvalues

The stability regions illustrated in section 3.2 suggest that no improvement can be expected with respect to the one-step method in case of complex or imaginary eigenvalues. In fig. 3.1-3.2 the imaginary stability boundary is in fact zero. Only if $c \geq 1$ we do have approximately the same region in the complex z-plane (compare [3], fig. 3.1). Numerical experiments with the characteristic roots and the parameter γ confirmed this presumption. Therefore it is advisable to use the two-step method only if the Jacobian of $H(t,U)$ has real eigenvalues.

3.5 The truncation error

We shall compare the truncation error of the two-step scheme generated by $\gamma = \gamma_R(c)$ and (3.1) with the truncation error of the one-step scheme generated by (3.2). For both schemes the truncation error is $O(\tau^4)$.

With $\theta_3 = 0$, the coefficients of the derivatives in the τ^4-term of the Taylor expansion of $U(t_{k+1})-L_k(U(t_{k-1}),U(t_k))$ are reduced to (compare [4], p. 19).

$$(3.18) \qquad \frac{1}{4} - (\gamma[\theta_1\lambda_{10}^3 + \theta_2(\lambda_{20}+\lambda_{21})^3] + (1-\gamma)\frac{c^4}{4}) ,$$

$$(3.19) \qquad \frac{1}{12} - (\gamma[\theta_2\lambda_{21}\lambda_{10}^2] + (1-\gamma)\frac{c^4}{12}) ,$$

$$(3.20) \qquad \frac{1}{8} - (\gamma[\theta_2(\lambda_{20}+\lambda_{21})\lambda_{21}\lambda_{10}] + (1-\gamma)\frac{c^4}{8}) ,$$

$$(3.21) \qquad \frac{1}{24} - (1-\gamma)\frac{c^4}{24} .$$

Substitution of the parameter values in (3.18)-(3.21) yields a measure for the truncation error. Thus, we arrive at table 3.2.

Table 3.2.

γ	(3.18)	(3.19)	(3.20)	(3.21)
$\gamma_R(.5)$	.1125	.0500	.0563	.0438
$\gamma_R(1)$	.2067	.0775	.1034	.0517
$\gamma_R(1.5)$	.2971	.1057	.1485	.0628
$\gamma_R(2)$	.3663	.1278	.1832	.0725
1	.0278	.0278	.0139	.0417

From this table we may conclude that the one-step scheme is more accurate than the two-step scheme.

4. The procedure two step runge kutta

4.1 General information

The programmed scheme is

$$(4.1)\quad \begin{cases} U_{k+1} = \gamma[U_k + (\beta_1-\beta_2^2/2\beta_3)r_k^{(0)} + (\beta_2^2/2\beta_3)r_k^{(2)}] + (1-\gamma)U_{k-1} , \\ r_k^{(0)} = \tau_k H(t_k , U_k) , \\ r_k^{(1)} = \tau_k H(t_k + (\beta_3/\beta_2)\tau_k , U_k + (\beta_3/\beta_2)r_k^{(0)}) , \\ r_k^{(2)} = \tau_k H(t_k + (2\beta_3/\beta_2)\tau_k , U_k + (2\beta_3/\beta_2)r_k^{(1)}) , \end{cases}$$

where $\gamma = \gamma_R(c)$.

The procedure uses the variable step-size mechanism as given in [4], p. 56-58. This mechanism is based on the last Taylor term taken into account. The formula for this term is

$$(4.3)\quad \begin{cases} th^3 dU_k = b_0\, r_k^{(0)} + b_2\, r_k^{(2)} + b_3\, r_k^{(3)} , \\ r_k^{(3)} = \tau_k H(t_k + \tau_k , U_{k+1}) , \\ b_0 = (2\beta_2\beta_3-\beta_2^2)/(12\beta_3^2 - 6\beta_2\beta_3) , \\ b_2 = \beta_2^2/(12\beta_3^2 - 6\beta_2\beta_3) , \\ b_3 = -2\beta_2\beta_3/(12\beta_3^2 - 6\beta_2\beta_3) . \end{cases}$$

In this formula a new evaluation of H(t,U) is used. If the integration step is accepted, we can use this evaluation in the next integration step for $r_k^{(0)}$. At each call of the procedure we have $\gamma = 1$, so we can start the integration pocess with one initial value.

4.2. Heading and parameters

```
procedure two step runge kutta(t, te, m0, m, u, derivative, k, kreject,
                               singlestep, sigma, step, tol, output);

integer     m0, m, k, kreject;
real        t, te, sigma, step, tol;
boolean     singlestep;
array       u;
procedure   derivative, output;
```

Parameters:

t : <variable>;
t represents the independent variable; when two step runge kutta is called, t should have its initial value to;

te : <expression>;
the end value of t;

m0, m : <expression>;
indices of the first and last equation of the system to be solved;

u : <array-identifier>;
the array u[m0:m] represents the numerical solution; when two step runge kutta is called, u should contain the initial vector $\tilde{U}_0$;

derivative : <procedure-identifier>;
derivative has to be declared by the user as follows:
procedure derivative (t,a);
real t; array a;
<replacement of the component a_i by the component $H_i(t,a_{m0},\ldots,a_m)$ for i=m0,...,m>;

k : <variable>;

k counts the number of integration steps, including the rejected ones;

kreject :<variable>;
kreject counts the number of rejected integration steps;

singlestep :<boolean expression>;
if singlestep is true, the procedure uses the one-step scheme generated by (3.2); otherwise the two-step scheme generated by (3.1); it is also possible to use expressions like t<t1, where t0<t1<te; in this case both schemes are used;

sigma :<expression>;
sigma denotes the spectral radius of the Jacobian; in each integration step the steplength tau satisfies the inequality tau $\leq \beta_{real}$ / sigma; if the spectral radius is not available, the user may substitute 0;

step :<variable>;
when two step runge kutta is called, step must be equal to the length of the first integration step to be executed; this initial step may be determined by accuracy considerations; after each integration step except the last one, step gives the step-size which has been used; at the end of the integration process step gives the step-size for a new first integration step for continuation of the integration;

tol :<expression>;
a measure of the required local accuracy; see the subprocedure test accuracy;

output :<procedure-identifier>;
output must be declared by the user;
the heading output is
procedure output;
<by this procedure, the user may order the values of t, k, kreject, step, u[m0],...,u[m] etc., to be printed; output may

also be used to stop the integration process, e.g. with the statement

if k > 1000 ∨ step < .01 then t:= te>;

4.3 The body of two step runge kutta

```
procedure two step runge kutta(t, te, m0, m, u, derivative, k,
kreject, singlestep, sigma, step, tol, output);
integer m0, m, k, kreject; real t, te, sigma, step, tol;
boolean singlestep; array u; procedure derivative, output;
begin integer j;
    real tau, tau0, b0, b2, b3, c, de, dem, int, mu, mu1, tl,
    tolint, ga, max, gamma, labda10, labda21, theta0, theta2;
    boolean first, last, reject, onestep;
    array ul, ull, k0, k1, k2[m0:m];

    procedure initialize;
    begin k:= kreject:= 0; ga:= sqrt(6);
        int:= te - t; tolint:= tol / int; tl:= t;
        max:= if singlestep then 2.5 / sigma else 4.3 / sigma;
        tau:= tau0:= if step < max then step else max;
        for j:= m0 step 1 until m do ul[j]:= u[j];
        onestep:= first:= reject:= true
    end initialize;


    procedure test growthparameter;
    begin max:= if onestep then 2.5 / sigma else 4.3 / sigma;
        if tau > max then tau:= max; c:= tau0 / tau;
        if c < .5 then
        begin c:= .5; tau:= tau0 / c end;
        last:= tau > te - tl; if last then
        begin step:= tau; tau:= te - tl; c:= tau0 / tau end;
        if c > 2 then onestep:= true
    end test growthparameter;


    procedure coefficient;
    if onestep then
    begin gamma:= 1; theta0:= .25; theta2:= .75;
        labda10:= 1 / 3; labda21:= 2 / 3;
        b0:= .5; b2:= - 1.5; b3:= 1
    end
    else if c = 1 then
    begin gamma:= 8 / (4 + ga); theta0:= - ga / 4;
        theta2:= - 2 × theta0; labda10:= ga / 12;
        labda21:= 2 × labda10; b2:= 2 / (1 - ga);
        b3:= - b2 / ga; b0:= - b2 - b3
    end
```

```
else
begin real c2, c3, c4, beta1, beta2, beta3, sum;
    c2:= c × c; c3:= c × c2; c4:= c × c3;
    sum:= 1.6 × (c + 0.75 × c2 + c3); gamma:= 1 +
    (sum − sqrt(sum × sum − 4 × c4)) / (2 × c4);
    beta1:= (1 + (1 − gamma) × c) / gamma;
    beta2:= (1 − (1 − gamma) × c2) / (2 × gamma);
    beta3:= (1 + (1 − gamma) × c3) / (6 × gamma);
    theta2:= beta2 × beta2 / (2 × beta3);
    theta0:= beta1 − theta2; labda10:= beta3 / beta2;
    labda21:= 2 × labda10;
    b2:= − 1 / ((6 − 12 × labda10) × labda10);
    b3:= − 2 × labda10 × b2; b0:= − b2 − b3
end coefficient;


procedure difference scheme;
begin if reject then
    begin for j:= m0 step 1 until m do k0[j]:= u1[j];
        derivative(t1, k0)
    end;
    for j:= m0 step 1 until m do
    k1[j]:= u1[j] + tau × labda10 × k0[j];
    t:= t1 + tau × labda10; derivative(t, k1);
    for j:= m0 step 1 until m do
    k2[j]:= u1[j] + tau × labda21 × k1[j];
    t:= t1 + tau × labda21; derivative(t ,k2);
    for j:= m0 step 1 until m do k1[j]:= u[j]:= gamma ×
    (u1[j] + tau × (theta0 × k0[j] + theta2 × k2[j])) +
    (1 − gamma) × u11[j];
    t:= if last then te else t1 + tau;
    derivative(t, k1); k:= k + 1
end difference scheme;


procedure test accuracy;
begin real discr, eps;
    dem:= 0; reject:= false;
    for j:= m0 step 1 until m do
    begin discr:= abs(tau ×
        (b0 × k0[j] + b2 × k2[j] + b3 × k1[j]));
        eps:= tolint × (abs(tau × k0[j]) + tau);
        reject:= discr > eps ∨ reject; de:= discr / eps;
        if de > dem then dem:= de
    end
end test accuracy;
```

```
procedure stepsize;
begin mu:= 1 / (1 + dem × dem) + .45; if reject then
    begin tau:= mu × tau; kreject:= kreject + 1;
        goto next level
    end;
    if ¬first then
    begin de:= mu × tau / tau0 + mu - mu1; tau0:= tau;
        tau:= de × tau; mu1:= mu
    end
    else
    begin tau0:= tau; tau:= mu × tau; first:= false;
        mu1:= mu
    end;
    if ¬last then step:= tau0
end stepsize;


procedure next integration step;
begin t1:= t;
    for j:= m0 step 1 until m do
    begin ull[j]:= ul[j]; ul[j]:= u[j]; k0[j]:= k1[j] end;
    onestep:= if singlestep then true else false;
    goto next level
end next integration step;

    initialize; output;
next level: test growthparameter; coefficient;
    difference scheme; test accuracy; stepsize; output;
    if t < te then next integration step;
    if ¬last then step:= tau
end two step runge kutta;
```

Next we discuss the several subprocedures which are used in two step runge kutta

The procedure initialize

In this procedure variables are initialized. At each call of two step runge kutta initialize is called once.

The procedure test growthparameter

In test growthparameter we check the steplength tau, the growthparameter c

and the place of the new integration point. If c > 2, the boolean onestep will have the value true and two step runge kutta uses the one-step scheme.

The procedure coefficient

In coefficient we compute:
the stability parameter γ,
the parameters θ_0, θ_2, λ_{10}, λ_{21} and
the parameters b_0, b_2, b_3.

The procedure difference scheme

In this procedure the components U[m0],...,U[m] of the numerical solution U_k are replaced by the components of the numerical solution U_{k+1}. At the same time $H(t_{k+1},U_{k+1})$ is computed.

The procedure test accuracy

In test accuracy the last Taylor term taken into account is computed for each component $U_k^{(i)}$ of U_k. If for some i_0

$$th^3 dU_k^{(i_0)} > \frac{h}{te-to} * (tol * H(t_k, U_k^{(i_0)}) + tol) ,$$

the last integration step is rejected.

The procedure stepsize

In stepsize the length of the next integration step is computed by means of the variable step-size mechanism mentioned in 4.1.

The procedure next integration step

In this procedure the components U[m0],...,U[m] of the numerical solution U_{k-1} are replaced by the components of the numerical solution U_k and the extra evaluation of $H(t_{k+1},U_{k+1})$ is utilized. Furthermore, the boolean singlestep is checked.

5 Numerical examples

In this chapter we present some results of the procedure two step runge kutta when applied to a number of differential equations with real eigenvalues. The one-step results were also obtained with two step runge kutta.

When the steplength is determined by a stability condition and not by a given accuracy condition, we may expect a great number of rejected integration steps, because the step-size mechanism is based on accuracy conditions. Therefore it is advisable to give the spectral radius of the system. The examples show that this is the best strategy for both methods when using two step runge kutta.

In the following subsections

k = the total number of executed integration steps,

k_{reject} = the number of rejected integration steps,

f.e. = the number of evaluations of H(t,U),

a.e. = $\max_{\{i,j\}} | \tilde{U}^{(i)}(t_j) - U_j^{(i)} |$, where the index i indicates the components of $\tilde{U}(t)$ and U and where $j=1,2,\ldots,k-k_{reject}$.

In the given figures the characters T and 0 refer to the two-step- and one-step method.

In section 5.3 an example of the procedure derivative and a call of two step runge kutta are given.

5.1 A stiff linear system

Consider the following initial value problem:

$$(5.1) \qquad \dot{U} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -5_{10}5 & -5_{10}5-15_{10}2 & -1501 \end{bmatrix} U, \; t \geq 0, \; U = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \; t = 0 .$$

The analytical solution is given by

$$(5.2) \qquad \tilde{U}(t) = e^{-t} \tilde{U}_0 .$$

The Jacobian has the eigenvalues $\delta_1 = -1000$, $\delta_2 = -500$ and $\delta_3 = -1$. Consequently we have stability if $\tau_k \leq \beta_{real} * 10^{-3}$.
First the predicted β_{real} was checked by using a uniform steplength τ from t = 0 to t = Kτ, where K is the number of integration steps. In this case $\beta_{real} = 4.5$. The results given in table 5.1 are a confirmation of the linear stability theory.

Table 5.1.

τ	K	a.e
.0045	200	$.1_{10}-7$
.0046	200	$.7_{10}13$

Secondly, (5.1) was integrated from t=0 to t=1 with variable steplength, both with sigma = 0 and sigma = 1000. For both the one-step- and two-step method, much better results were obtained with sigma = 1000. In this case, we have no rejected steps; the steplength is completely determined by the severe stability condition. The results are given in table 5.2.

Table 5.2.

	k		k_{reject}		f.e.		a.e.	
sigma	0	1000	0	1000	0	1000	0	1000
	480	401	133	0	1573	1203	$.2_{10}-2$	$.3_{10}-7$
one-	512	401	154	0	1690	1203	$.1_{10}-3$	$.3_{10}-7$
step	503	401	148	0	1657	1203	$.1_{10}-4$	$.3_{10}-7$
	517	401	152	0	1703	1203	$.1_{10}-5$	$.3_{10}-7$
	369	234	115	0	1222	702	$.2_{10}-2$	$.4_{10}-7$
two-	383	234	115	0	1264	702	$.2_{10}-3$	$.4_{10}-7$
step	381	234	113	0	1256	702	$.2_{10}-4$	$.3_{10}-7$
	399	234	123	0	1320	702	$.2_{10}-5$	$.4_{10}-7$

5.2 A simple non-linear equation

Consider the initial value problem

$$(5.3) \qquad \begin{cases} \dot{U} = 100 - U^2 , & t > 0 , \\ U = 0 , & t = 0 , \end{cases}$$

with the analytical solution

$$(5.4) \qquad \tilde{U}(t) = 10 - 20/(e^{20t}+1) .$$

We have integrated (5.3) on the interval [0,10]. On this integration interval the spectral radius comes up to 20. For sigma = 0 and sigma = 20 we have almost the same number of rejected steps. This means that on a part of the integration interval, the steplength is strongly determined by accuracy conditions. The reason for this is that near the origin, the derivative is very large. In general in this situation, the one-step results are slightly better. The results are presented in table 5.3 and fig. 5.1-5.2.

Table 5.3

	k		k_{reject}		f.e.		a.e.	
sigma	0	20	0	20	0	20	0	0
one-step	104	84	22	1	334	253	$.4_{10}-0$	$.4_{10}-0$
	119	87	37	3	394	264	$.3_{10}-0$	$.3_{10}-0$
	123	106	32	7	401	325	$.4_{10}-2$	$.4_{10}-2$
	199	161	47	7	644	490	$.7_{10}-4$	$.7_{10}-4$
	362	336	36	8	1122	1026	$.4_{10}-5$	$.4_{10}-5$
two-step	86	57	31	4	289	175	$.5_{10}-0$	$.5_{10}-0$
	91	58	25	3	298	177	$.2_{10}-0$	$.2_{10}-0$
	105	80	27	7	342	247	$.9_{10}-2$	$.9_{10}-2$
	162	134	30	7	516	409	$.5_{10}-3$	$.5_{10}-3$
	346	312	35	10	1073	946	$.3_{10}-4$	$.3_{10}-4$

5.3 A problem in nuclear reactor physics

In nuclear reactor physics the following system is of interest

$$(5.5)\qquad \begin{cases} \dot{U}_1 = .2(U_2-U_1) & t \geq 0 , \\ \dot{U}_2 = 10U_1 - (60+.125t)U_2 + .124t , & t \geq 0 , \\ U_1 = U_2 = 0 , \ t = 0 . \end{cases}$$

Because an analytical solution was not obtained, we use the reference solution

$$(5.6)\qquad \begin{cases} \bar{U}_1 = .01248223537 , & t = 10, \\ \bar{U}_2 = .02224529798 , & t = 10. \end{cases}$$

On the integration interval [0,10] the eigenvalues are approximately -60 and -.17. The results of integration are presented in table 5.4 and fig. 5.4-5.5. From the given numbers for k_{reject}, we see that the steplength is determined by the stability condition.

In table 5.4 and fig. 5.4-5.5 $\varepsilon = \max_{i=1,2} |\bar{U}_i - U_i|$ in t=10.

Table 5.4

	k		k_{reject}		f.e.		ε	
sigma	0	60	0	60	0	60	0	60
	308	242	84	0	1008	726	$.8_{10}-6$	$.6_{10}-4$
one-	323	243	95	1	1064	730	$.2_{10}-5$	$.3_{10}-6$
step	348	244	108	1	1152	733	$.4_{10}-6$	$.4_{10}-6$
	347	256	109	6	1150	774	$.1_{10}-8$	$.5_{10}-7$
	244	141	79	0	811	423	$.6_{10}-4$	$.5_{10}-8$
two-	236	142	69	1	779	427	$.9_{10}-6$	$.7_{10}-8$
step	235	149	75	4	780	450	$.3_{10}-6$	$.6_{10}-8$
	266	160	83	7	881	487	$.2_{10}-6$	$.6_{10}-8$

Next the procedure derivative and a call of two step runge kutta are presented.

```
procedure derivative (t,a); real t; array a;
begin real a1, a2;
      a1:= a[1]; a2:= a[2];
      a[1]:= .2 * (a2-a1);
      a[2]:= 10 * a1 - (60+.125 t) * a2 + .124 * t
end;
t:= 0; step:= .05; u[1]:= u[2]:= 0;
two step runge kutta (t, 10, 1, 2, u, derivative, k, kreject,
                      false, 60, step, 10-2, output);
```

5.4 An example from the literature

The last example we consider is the initial value problem (compare [5], p. 286)

$$(5.7)\qquad \begin{cases} \dot{U} = -20(U-F(t)) + \dot{F}(t), \\ U = 10\ ,\ t = 0\ , \\ F(t) = 10 - (10+t)e^{-t}\ , \end{cases}$$

with the analytical solution

$$(5.8)\qquad U(t) = F(t) + 10e^{-20t}\ ,$$

The solution (5.8) contains a rapidly decaying component and a slowly decaying component. The eigenvalue is -20 and the solution is desired from t=0 to t=20. Thus, the component exp(-20t) soon becomes negligible compared to the exp(-t) component. The maximum error, presented in table 5.5 and fig. 5.5-5.6, represents the error early in the integration. This error will decrease late in the integration.

Using the one-step method early in the integration and the two-step method late in the integration would yield results comparable to those of the two-step method.

Table 5.5

	k		k_{reject}		f.e.		a.e.	
sigma	0	20	0	20	0	20	0	20
one-step	209	165	50	2	677	497	$.3_{10}-0$	$.3_{10}-0$
	244	170	80	4	812	514	$.2_{10}-0$	$.2_{10}-0$
	289	249	75	26	942	773	$.1_{10}-2$	$.1_{10}-2$
	486	441	105	56	1563	1379	$.8_{10}-4$	$.8_{10}-4$
	1020	983	202	162	3262	3111	$.1_{10}-4$	$.1_{10}-4$
two-step	160	100	53	2	533	302	$.6_{10}-0$	$.6_{10}-0$
	195	136	55	18	640	426	$.2_{10}-0$	$.2_{10}-0$
	235	206	28	7	733	625	$.2_{10}-2$	$.2_{10}-2$
	413	398	70	58	1309	1252	$.2_{10}-3$	$.2_{10}-3$
	888	877	151	141	2815	2772	$.4_{10}-4$	$.4_{10}-4$

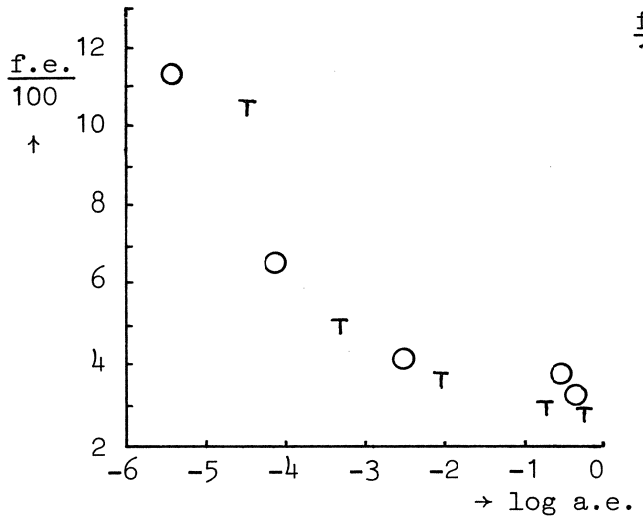

Fig. 5.1. sigma = 0

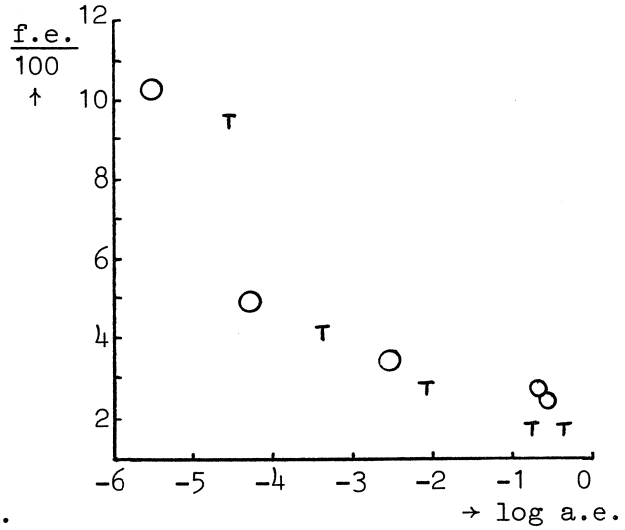

Fig. 5.2. sigma = 20

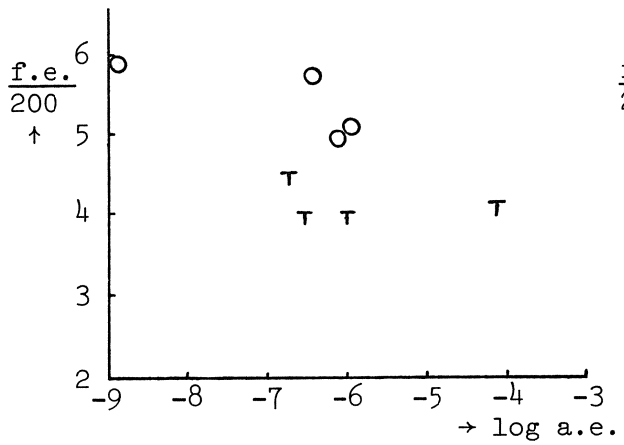

Fig. 5.3. sigma = 0

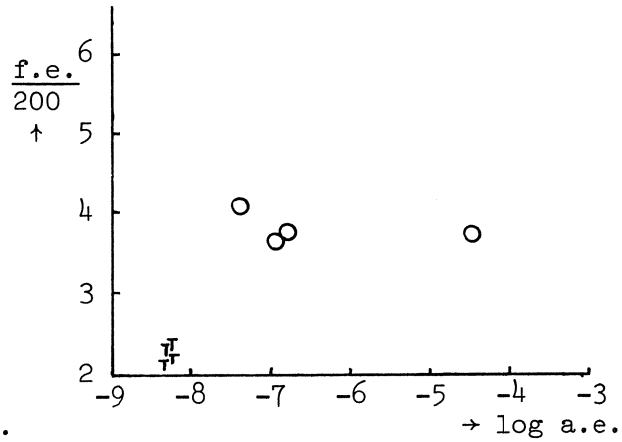

Fig. 5.4. sigma = 60

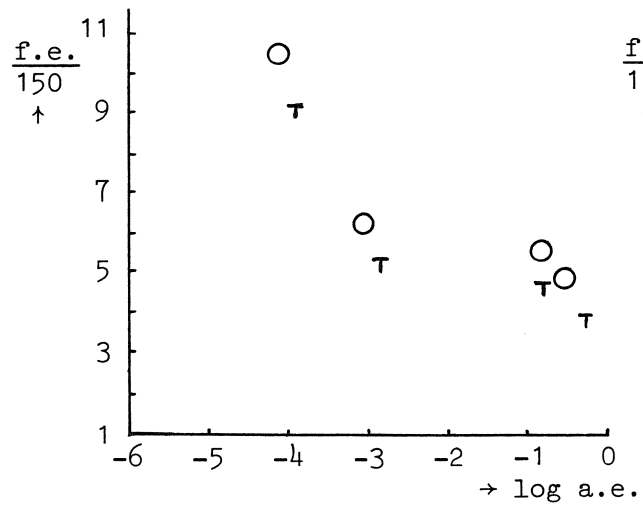

Fig. 5.5. sigma = 0

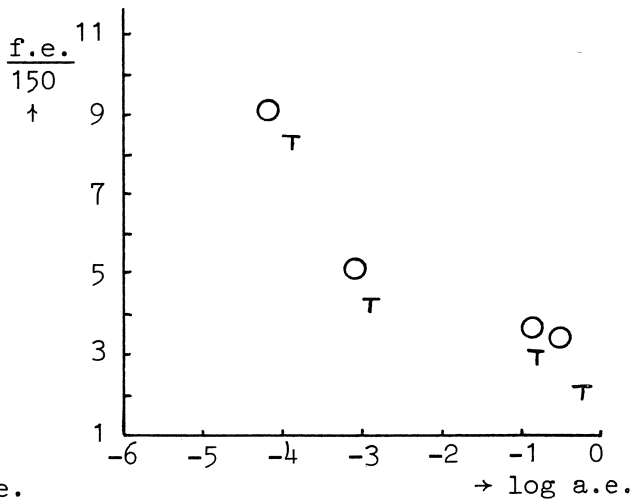

Fig. 5.6. sigma = 20

References

[1] Houwen, P.J. van der, A note on two-step Runge-Kutta methods, Report TN 61/71, Mathematisch Centrum, Amsterdam, (1971).

[2] Houwen, P.J. van der, Stabilized Runge-Kutta methods with limited storage requirements. Report TW 124/71, Mathematisch Centrum, Amsterdam, (1971).

[3] Houwen, P.J. van der, One-step methods for linear initial value problems I. Polynomial methods, Report TW 119, Mathematisch Centrum, Amsterdam, (1970).

[4] Zonneveld, J.A., Automatic numerical integration, MC Tract 8, Mathematisch Centrum, Amsterdam, (1964).

[5] Lapidus, L. and Seinfeld, J.H., Numerical integration of ordinary differential equations, Academic Press, New York, London, (1971).

[6] Byrne, G.D. and R.J. Lambert, Pseudo-Runge-Kutta methods involving two points, J. Assoc. Comput. Mach. 13, p. 114 (1966).